Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О.В. Непомнящий
подпись инициалы, фамилия
«____» _____ 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Модуль голосовой идентификации диктора
Тема

09.04.01 Информатика и вычислительная техника
код и наименование направления

09.04.01.01 Высокопроизводительные вычислительные системы
код и наименование магистерской программы

Научный руководитель	_____	доцент, канд. техн. наук	М.С. Медведев
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		И.В. Дашкевич
	подпись, дата		инициалы, фамилия
Рецензент	_____		
	подпись, дата	должность, ученая степень	инициалы, фамилия
Нормоконтролер	_____		В.И. Иванов
	подпись, дата		инициалы, фамилия

Красноярск 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Системы голосовой идентификации диктора (СГИД) быстро развиваются в последнее время. Причиной развития СГИД это их востребованность в таких областях, как биометрический поиск, голосовая верификация пассажиров и водителя, разграничение прав доступа к информации с помощью голосовой биометрии и т. д. [1]. Важным достоинством СГИД по отношению к другим биометрическим системам идентификации является их дешевизна. Важно также, что современные СГИД по уровню надёжности идентификации не уступают, а иногда и превосходят, к примеру, системы идентификации человека по изображению [2].Эволюция систем распознавания речи привела к созданию интеллектуальных систем, позволяющих не только распознавать, но и автоматически синтезировать человеческую речь.

Несмотря на уникальность голоса человека, ни одна из СГИД, как и любая другая биометрическая система, не может гарантировать 100% надёжность идентификации. Основными источниками ошибок в СГИД являются: окружение (шум, реверберация и т.д.); особенности речи (длительность, тональность, уровень голосового усилия и т.д.); канал связи (искажения микрофона и канала передачи, погрешности кодирования аудио сигнала и т.д.) [2].

В общем случае идентификация личности по голосу требует решения большого числа разнородных задач, основными из которых являются следующие:

- выделение вокализованных участков аудио сигнала путём отбрасывания пауз и участков, содержащих различного рода помехи;
- разделение речи дикторов (задача диаризации);
- выделение характерных признаков голоса диктора.

Применение технологии идентификации и верификации диктора позволяет:

- осуществить тексто- и языконезависимую идентификацию диктора;
- выделить из общего объёма данных звуковые файлы, содержащие речь интересующего диктора;
- подтвердить принадлежность звуковых данных тому или иному диктору;
- снизить риск пропуска файла с речью диктора из-за перегруженности операторов;
- обработать большое количество речевой информации и подготовить данные для дальнейшей обработки оператором [14].

ЗАДАЧИ

- Разработать метод выделения признаков речевого сигнала, позволяющий проводить идентификацию дикторов;
- проанализировать существующие подходы и выбрать алгоритм классификации пользователей по их голосовым характеристикам;
- разработать модуль голосовой идентификации диктора на рабочей среде MATLAB, в дальнейшем подключаемого к системе распознавания речи, с целью повышения качества распознавания;
- внедрить дополнительный этапа классификации диктора в систему распознавания речи — повышение качества распознавания речи за счёт выбора оптимального классификатора, обученного на речевом материале, с схожими с диктором голосовыми характеристиками;
- определить качество распознавания модуля идентификации диктора.

1 Анализ системы

1.1 Анализ технического задания

Распознавание дикторов объединяет идентификацию и верификацию дикторов.

Идентификация диктора — процесс, выявляющий личность по образцу голоса путём сравнения данного образца с образцами, сохранёнными в базе. Результатом процесса идентификации является список кандидатов. Выполняющая система может выдать список фиксированного размера либо приять решение о включении пользователя в список кандидатов на основании заданного критерия. Если предусмотрена возможность того, что в процессе идентификации участвует пользователь, не зарегистрированный в системе, то говорят об идентификации на открытом множестве. Если все пользователи, которые проходят процедуру идентификации, зарегистрированы в системе, то говорят об идентификации на замкнутом множестве.

Верификация диктора — процесс, при котором с помощью сравнения представленного образца с хранящимся в базе шаблоном проверяется запрошенная идентичность. Результатом верификации является положительное либо отрицательное решение. Иногда используется термин ≪обнаружение по голосу≫(speaker detection [4]). В задаче обнаружения применяются несколько иные термины и приоритеты, но, по сути, верификация и обнаружение являются одной той же задачей [3].

В нашей работе большое внимание отведено задаче выделения характерных признаков голоса (ХПГ). Вообще говоря, для выделения ХПГ можно использовать спектрально-формантный анализ, статистические характеристики основного тона голоса, параметрическое представление

аудиосигнала. ХПГ можно искать в амплитуде и диапазоне частот звукового сигнала, а также в его спектре. Широкий список публикаций, посвящённых перечисленным методам, представлен в работе [5].

Существует два способа распознавания аудио-сигнала, дикторозависимый и дикторонезависимый.

Дикторозависимая система предназначена для использования одним пользователем, в то время как дикторонезависимая система предназначена для работы с разными дикторами. Дикторонезависимость – труднодостижимая цель, так как при обучении системы, она настраивается на параметры того диктора, на примере которого обучается. Частота ошибок распознавания таких систем обычно в 3-5 раз больше, чем частота ошибок дикторозависимых систем [9].

1.2 Средства разработки

Среда разработки Matlab R15b

В качестве платформы для исследований и разработки модуля голосовой идентификации диктора был выбран пакет MATLAB. Данный выбор объясняется следующим:

MATLAB - пакет прикладных программ для решения задач сложных технических вычислений, а также используемый в этом пакете язык программирования. MATLAB используют более 1 000 000 научных и инженерных работников, он работает на большинстве современных операционных систем, включая GNU/Linux, Mac OS, Solaris и Microsoft Windows.

Язык MATLAB является высокоуровневым интерпретируемым языком программирования, включающим основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования.

MATLAB предоставляет пользователю большое количество (несколько сотен) функций для анализа данных, использующиеся практически во все областях математики.

MATLAB - это удобные средства для разработки алгоритмов, включающие высокоуровневые, с использованием концепций объектно-ориентированного программирования. В нём имеются все необходимые средства интегрированной среды разработки, включая отладчик и профайлер. Функции для работы с целыми типами данных облегчают создание алгоритмов для микроконтроллеров и других приложений, где это необходимо.

Встроенная среда разработки позволяет создавать графические интерфейсы пользователя с различными элементами управления, такими как кнопки, поля ввода и другими. С помощью компонента MATLAB Compiler

эти графические интерфейсы могут быть преобразованы в самостоятельные приложения, для запуска которых на других компьютерах необходима установленная библиотека MATLAB Component Runtime.

В пакет MATLAB входят различные интерфейсы для получения доступа к внешним подпрограммам, написанным на других языках программирования, данным, клиентам и серверам, общающимся через технологии Component Object Model или Dynamic Data Exchange, а также периферийным устройствам, которые взаимодействуют напрямую с MATLAB. Многие из этих возможностей известны под названием MATLAB API.

Система Matlab предоставляет мощный язык программирования, ориентированный на математические преобразования, который превосходит по возможности и скорости вычислений традиционные языки программирования [8].

Для решения проблемы классификации, сообщающиеся функции и процедуры системы собираются в специальные папки. Это создаёт концепцию пакетов прикладных программ, представляющих собой коллекции М-файлов для решения определённой задачи или проблемы. Именно пакеты прикладных программ – Matlab Application Toolboxes, которые входят в состав семейства продуктов Matlab, позволяют находиться этой системе на уровне самых современных приложений.

Для моделирования структуры нейронной сети выбран пакет Neural Networks был. В пакет входят более полутора десятка известных типов искусственных нейронных сетей и обучающих правил, которые позволяют пользователю выбрать наиболее подходящую для конкретного приложения или исследовательской задачи парадигму. Для каждого типа архитектуры и обучающего алгоритма имеются функции инициализации, обучения, адаптации, создания и моделирования, демонстрации и примеры применения.

Один из пакетов системы - Wavelet Toolbox, предоставляет разнообразные возможности обработки сигналов с помощью вейвлетов. С его помощью реализуются принципиально новые виды декомпозиции и реконструкции сигналов и изображений с повышенной эффективностью и новыми качественными возможностями – например, в идентификации тонких локальных особенностей функций и сигналов.

В стандартный пакет аудио поддержки системы Matlab R15b входят функции, которые позволяют произвести запись звукового сигнала с возможностью настройки значений частоты дискретизации и разрядности.

Встроенная в Matlab среда разработки пользовательского интерфейса Matlab GUIDE позволяет реализовать элементы визуально-ориентированного программирования (инструментальные панели, кнопки, меню и т.д.). В системе Matlab присутствует уникальная возможность сохранения значений всех обрабатываемых переменных в постоянную память из оперативной. Переменные, которые сохранены в специальных mat-файлы, могут быть в дальнейшем загружены т.к. одновременно-файл могут быть сохранены переменные разных типов и структура самого хранения является упорядоченной, то данную возможность системы Matlab удобнее применять для создания базы данных эталонов.

1.3 Методы выделения признаков речевого сигнала

Существует несколько способов преобразования сигнала, о самых распространённых из них будет описано ниже.

1.3.1 Преобразование Фурье

Преобразование Фурье — это функция, которая описывает амплитуду и фазу каждой синусоиды, соответствующей определённой частоте. (Амплитуда представляет высоту кривой, а фаза представляет начальную точку синусоиды). Эта новая функция, описывающая коэффициенты («амплитуды») при разложении исходной функции на элементарные составляющие — гармонические колебания с разными частотами (подобно тому, как музыкальный аккорд может быть выражен в виде амплитуд нот, которые его составляют).

Преобразование Фурье функции f вещественной переменной является интегральным и задаётся с помощью следующей формулы:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ix\omega}\, dx. \quad , \qquad (1.1)$$

Хотя формула, которая задаёт преобразование Фурье, имеет понятный смысл только для функций класса $L_1(\mathbb{R})$, преобразование Фурье может использоваться и для более широкого класса функций и даже обобщённых функций. Это возможно благодаря особому ряду свойств преобразования Фурье:

Преобразование Фурье является линейным оператором:

$$\widehat{(\alpha f + \beta g)} = \alpha \hat{f} + \beta \hat{g}. \qquad (1.2)$$

Справедливо равенство Парсеваля: если $f \in L_1(\mathbb{R}) \cap L_2(\mathbb{R})$, то преобразование Фурье сохраняет L_2-норму:

$$\int_{-\infty}^{\infty} |f(x)|^2\,dx = \int_{-\infty}^{\infty} |\hat{f}(w)|^2\,dw. \tag{1.3}$$

Это свойство позволяет по непрерывности распространить определение преобразования Фурье на всё пространство $L_2(\mathbb{R})$. Равенство Парсеваля будет при этом справедливо для всех $f \in L_2(\mathbb{R})$.

Формула обращения:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(w) e^{ix\omega}\,dw \quad , \tag{1.4}$$

справедлива, если интеграл в правой части имеет смысл. В частности, это верно, если функция f является достаточно гладкой. Если $f \in L_2(\mathbb{R})$, то формула также верна, поскольку равенство Парсеваля позволяет придать интегралу в правой части смысл с помощью предельного перехода.

Эта формула объясняет физический смысл преобразования Фурье: правая часть — (бесконечная) сумма гармонических колебаний $e^{i\omega x}$ с частотами ω, амплитудами $\frac{1}{\sqrt{2\pi}}|\hat{f}(\omega)|$ и фазовыми сдвигами $\arg \hat{f}(\omega)$ соответственно.

Теорема о свёртке: если $f,\ g \in L_1(\mathbb{R})$, тогда

$$\begin{aligned} &\widehat{(f * g)} = \sqrt{2\pi}\,\hat{f}\hat{g}, \\ &(f * g)(t) = \int_{-\infty}^{\infty} f(t-s)g(s)\,ds. \end{aligned} \tag{1.5}$$

Эта формула может быть распространена и на случай обобщённых функций.

Преобразование Фурье и дифференцирование. Если $f,\ f' \in L_1(\mathbb{R})$, то

$$\widehat{(f')} = i\omega\widehat{f}. \tag{1.6}$$

Из этой формулы легко выводится формула для n-й производной:

$$\widehat{(f^{(n)})} = (i\omega)^n \widehat{f}. \tag{1.7}$$

Формулы верны и в случае обобщённых функций.

Преобразование Фурье и сдвиг.

$$\widehat{f(x - x_0)} = e^{-i\omega x_0}\hat{f}(w). \tag{1.8}$$

Эта и предыдущая формула являются частными случаями теоремы о свёртке, так как сдвиг по аргументу — это свёртка со сдвинутой дельта-функцией $\delta(x - x_0)$, а дифференцирование — свёртка с производной дельта-функции.

Преобразование Фурье и растяжение.

$$\widehat{f(ax)} = |a|^{-1}\hat{f}(w/a). \tag{1.9}$$

Преобразование Фурье обобщённых функций. Преобразование Фурье можно определить для широкого класса обобщённых функций. Определим вначале пространство гладких быстро убывающих функций (пространство Шварца):

$$S(\mathbb{R}) := \left\{\varphi \in C^\infty(\mathbb{R}) : \forall n,\ m \in \mathbb{N}\ x^n\varphi^{(m)}(x) \xrightarrow{x\to\infty} 0\right\}. \tag{1.10}$$

Ключевым свойством этого пространства является то, что это инвариантное подпространство по отношению к преобразованию Фурье [7].

1.3.2 Вейвлет-преобразование

Широко используемое преобразование Фурье для анализа сигналов, как непрерывных, так и дискретных, оказывается недостаточно эффективным при обработке сложных сигналов. Например, Фурье спектры для сигналов из двух синусоид, которые с разными частотами, первый из которых, представляющий собой сумму синусоид, а второй представляет собой, последовательно следующие друг за другом синусоиды, одинаковы и будут выглядеть как два пика на двух фиксированных частотах (рисунок 1). Из этого следует, преобразование Фурье в своём обычном виде не приспособлено для анализа нестационарных сигналов, так как теряется информация о временных характеристиках сигнала. Речевой сигнал является примером нестационарного процесса, в котором информативным является сам факт изменения его частотно-временных характеристик.

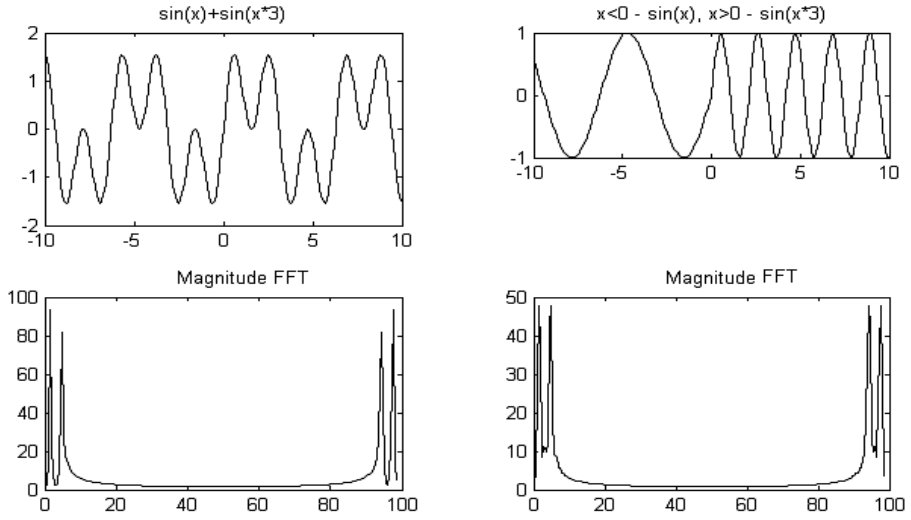

Рисунок 1– Пример неинформативности преобразования Фурье

Для анализа таких процессов требуются базисные функции, способные выявлять в исследуемом сигнале как частотные, так и его временные

характеристики, т. е. функции со свойствами частотно-временной локализации. Такие возможности предоставляют вейвлеты, являющиеся обобщением спектрального анализа.

Вейвлеты – функции двух аргументов – масштаба и сдвига. В отличие от стандартного преобразования Фурье, они позволяют обрабатывать сигнал одновременно в физическом пространстве – время, координата, и частотном пространстве

$$\left[\mathrm{W}_{\psi} f\right](x,a) = |a|^{-1/2} \int_{-\infty}^{+\infty} f(t)\psi\left(\frac{t-x}{a}\right)dt, \qquad (1.11)$$

здесь $\psi\left(\frac{t-x}{a}\right)$ – вейвлет, a – масштабный коэффициент, x – параметры сдвига. Таким образом, вейвлет-преобразование обеспечивает двумерное представление исследуемого сигнала в частотной области в плоскости частота-положение. Аналогом частоты при этом является масштаб аргумента базисной функции (чаще всего – времени), а положение характеризуется её сдвигом. Это позволяет найти особенности сигналов, одновременно локализуя их на временной шкале. Другими словами, вейвлет-анализ можно охарактеризовать как спектральный анализ локальных возмущений.

В теории вейвлет-анализа существует множество направлений. Например, используя многомасштабный (кратномасштабный) вейвлет-анализ сигнал можно представить как последовательность образов с разной степенью детализации, что позволяет найти локальные особенности сигнала и классифицировать их по интенсивности.

Анализ основывается на разложении сигнала по функциям, образующим ортонормированный базис. Каждую функцию можно разложить на некотором заданном уровне разрешения (масштабе) j_n в ряд вида:

$$f(x) = \sum_{k=0}^{2M-1} s_{j_n,k}\varphi_{j_n,k} + \sum_{j\geq j_n}^{j_{\max}} \sum_{k=0}^{2M-1} d_{j_n,k}\psi_{j,k}, \quad (1.12)$$

где $\varphi_{j_n,k}$ и $\psi_{j,k}$ – масштабированные и смещённые версии скейлинг-функции (масштабной функции) φ и "материнского вейвлета" ψ;

$s_{j,k}$ – коэффициенты аппроксимации;

$d_{j,k}$ – детализирующие коэффициенты.

На рисунке 2-5 представлены аналитические графики функций φ и ψ указанных вейвлетов.

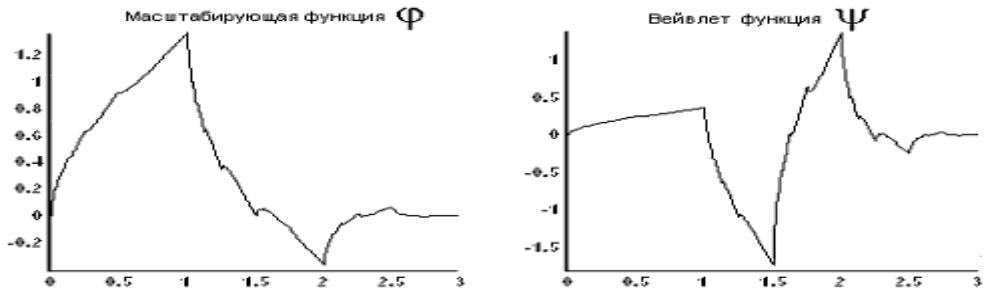

Рисунок – 2 Аналитические графики функций φ и ψ, вейвлет Симлета

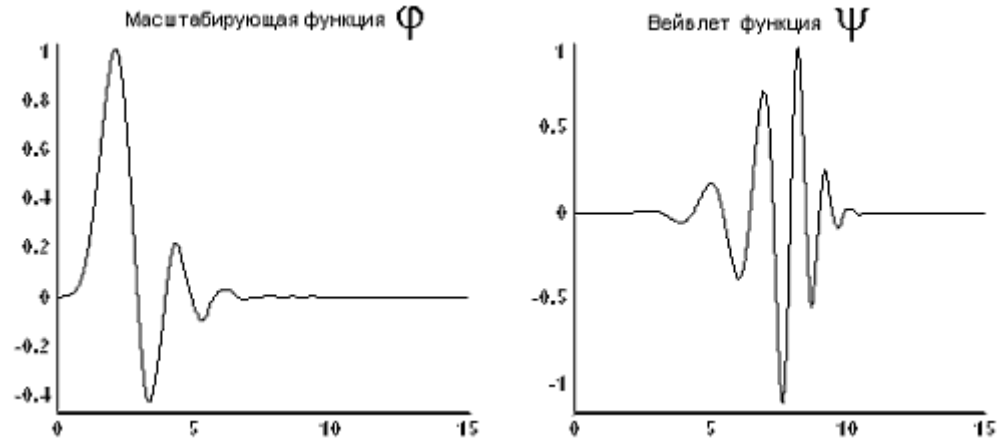

Рисунок – 3 Аналитические графики функций φ и ψ, вейвлет Добеши 8

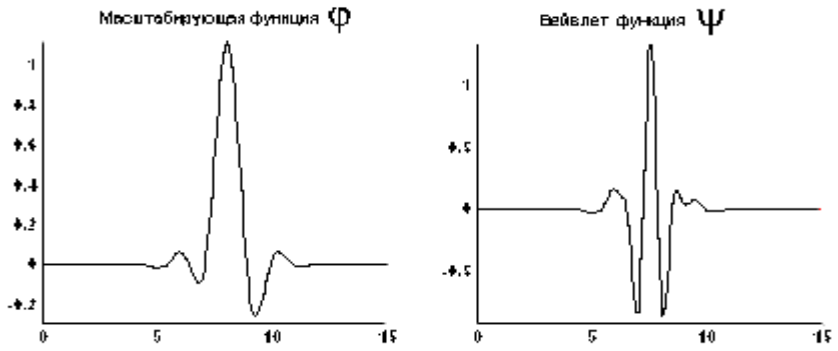

Рисунок – 4 Аналитические графики функций φ и ψ, вейвлет Симлета 8

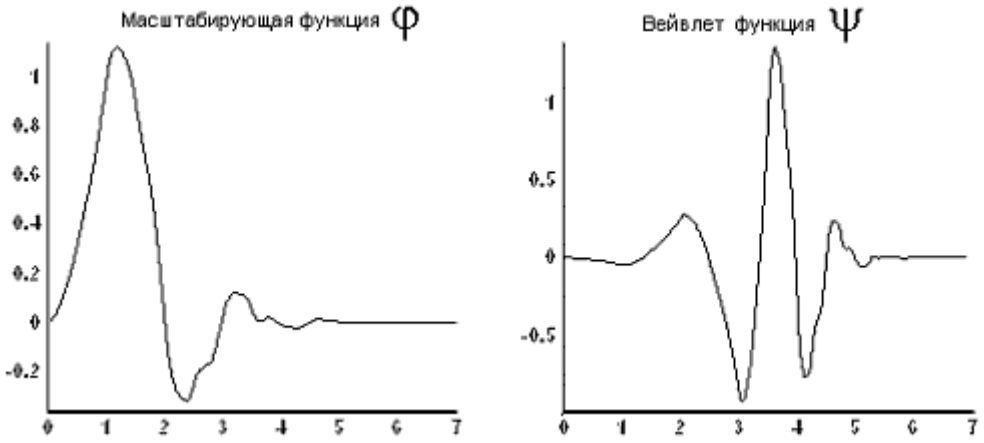

Рисунок 5- Аналитические графики функций φ и ψ, вейвлет Добеши 4

Масштабирование и смещение функций φ и ψ находится по законам

$$\varphi_{j,k} = 2^{j/2}\varphi(2^j x - k),$$
$$\psi_{j,k} = 2^{j/2}\psi(2^j x - k). \quad , \qquad (1.13)$$

В свою очередь сами функции φ и ψ определяются как:

$$\varphi(x) = \sqrt{2}\sum_{k=0}^{2M-1} h_k \varphi(2x-k)$$

$$\psi(x) = \sqrt{2}\sum_{k=0}^{2M-1} g_k \varphi(2x-k) \quad , \qquad (1.14)$$

где $g_k = (-1)^k h_{2M-k-1}$

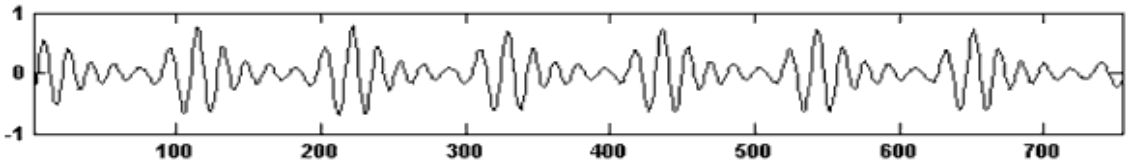

Рисунок – 6 Скелет максимумов фонемы "a", линия коэффициентов вейвлет-преобразования фонемы "a",

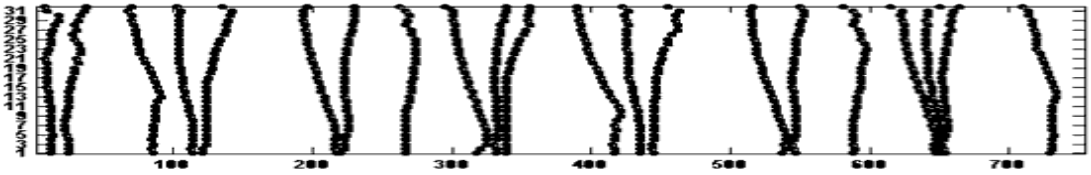

Рисунок – 7 Скелет максимумов фонемы "a":
скелет максимумов фонемы "a"

Выполняя свойства ортогональности масштабных функций и задавая значения M, можно вычислять конкретные значения коэффициентов hk, определяющие ортогональные вейвлеты. Например, задав M=2 получаем ряд коэффициентов hk, определяющий вейвлет Добеши 4.

Таким образом, ортогональный вейвлет-анализ сводится к нахождению коэффициентов аппроксимации $s_{j,k}$ и детализирующих коэффициентов $d_{j,k}$ в разложении сигнала f(x) по формуле (1.11). Выше приведён пример скелета максимумов для фонемы "a" (рисунок 6,7).

Множество точек на (x, a), в которых находятся локальные "пики" вейвлет-преобразования, образуют скелет максимумов. Этих точек часто очень много в области малых масштабов. Их появлением вейвлет-преобразование реагирует на любые негладкости сигнала. При росте масштаба мелкие негладкости исчезают, а вместе с ними и точки максимумов. Оставшиеся сливаются в довольно гладкие кривые, которые при дальнейшем росте масштаба тоже сливаются друг с другом. При этом они либо "аннигилируют", либо продолжают "расти" в область ещё более крупных масштабов, и т. д. В определённом смысле, вся существенная информация о сигнале находится в значениях скелета максимумов вейвлет-преобразования (рисунок 6). Использование этого механизма позволяет более эффективно выделять фонемы из речевого потока.

Достоинства:

- Вейвлетные преобразования обладают всеми достоинствами преобразований Фурье.
- Вейвлетные базисы могут быть хорошо локализованными как по частоте, так и по времени. При выделении в сигналах хорошо локализованных разномасштабных процессов можно рассматривать только те масштабные уровни разложения, которые представляют интерес.
- Базисные вейвлеты могут реализоваться функциями различной гладкости.

Недостатки:

- Можно выделить один недостаток, это относительная сложность преобразования.

1.3.3 Преобразование Гильберта-Хуанга

Под преобразованием Гильберта-Хуанга (Hilbert-Huang transform – HHT) понимается метод эмпирической модовой декомпозиции (EMD) нелинейных и нестационарных процессов и Гильбертов спектральный анализ (HSA) [6]. Этот метод потенциально жизнеспособный для нелинейного и нестационарного анализа данных, специально для частотно-энергетических временных представлений.

EMD-HSA предложил Норден Хуанг в 1995 в США (NASA) для изучения поверхностных волн тайфунов, включая возможность на анализ произвольных временных рядов коллективом соавторов в 1998 г. [6,7]. В последующие годы, активно расширяя применения алгоритма для других новых отраслей науки и техники, взамен термина EMD-HSA был принят более короткий термин преобразования HHT.

EMD (Empirical Mode Decomposition) – метод разложения сигналов на функции, получившие названия внутренних или «эмпирических мод». Метод представляет собой адаптивную итерационную вычислительную процедуру разложения исходных данных (непрерывных или дискретных сигналов) на эмпирические моды или внутренние колебания.

Огибающие сигналов. У каждого сигнала присутствуют локальные экстремумы: чередующиеся локальные максимумы и локальные минимумы с произвольным расположением по координатам (независимым переменным) сигналов. По этим экстремумам с использованием методов аппроксимации можно построить две огибающие сигналов: нижнюю – построенную по точкам локальных минимумов, и верхнюю – построенную по точкам локальных максимумов, а также функцию «среднего значения огибающих», которой отвечает срединная линия, расположенная в точности между нижней и верхней огибающими [17].

Функции внутренних мод сигналов. Модовая декомпозиция сигналов

основана на предположении, что любые данные состоят из различных внутренних колебаний (intrinsic mode functions, IMF). В любой момент времени данные могут иметь множество сосуществующих внутренних колебаний – IMFs. Каждое колебание, линейное или нелинейное, представляет собой модовую функцию, которая имеет экстремумы и нулевые пересечения. Кроме того, колебания в определенной степени «симметричны» относительно локального среднего значения. Конечные сложные данные образуются суммой модовых функций, наложенных на региональный тренд сигнала.

Эмпирическая мода – это такая функция, которая обладает следующими свойствами:

1. Количество экстремумов функции (максимумов и минимумов) и количество пересечений нуля не должны отличаться более чем на единицу.
2. В любой точке функции среднее значение огибающих, определенных локальными максимумами и локальными минимумами, должно быть нулевым.

IMF представляет собой колебательный режим, но вместо постоянной амплитуды и частоты, как в простой гармонике, у IMF могут быть переменная амплитуда и частота, как функции независимой переменной (времени, координаты, и пр.). Первое свойство гарантирует, что локальные максимумы функции всегда положительны, локальные минимумы соответственно отрицательны, а между ними всегда имеют место пересечения нулевой линии. Второе свойство гарантирует, что мгновенные частоты функции не будут иметь нежелательных флуктуаций, являющихся результатом асимметричной формы волны.

Любую функцию и любой произвольный сигнал, изначально содержащие произвольную последовательность локальных экстремумов (минимум 2), можно разделить на семейство функций IMFs и остаточный тренд. Если данные лишены экстремумов, но содержат точки перегиба

(«скрытые» экстремумы наложения модовых функций и крутых трендов), то для открытия экстремумов может использоваться дифференцирование сигнала.

Схема преобразования Гильберта-Хуанга может быть разделена на две части. В первом этапе, экспериментальные данные разлагаются в ряд внутренних модовых функций (IMFs). Эта декомпозиция рассматривается как расширение данных в терминах внутренних модовых функций. Иначе, эти внутренние модовые функции представлены как базис преобразования, которое может быть линейным или нелинейным, как диктуется по условиям. Так как IMFs имеют хорошие Гильбертовы преобразования, то могут быть вычислены соответствующие мгновенные частоты. Таким образом, в следующем шаге мы можем с лёгкостью локализовать любое явление, как во времени, так и на частотной оси. Локальная энергия и мгновенная частота, выведенная из IMFs, дают нам дистрибутивные “энергетические время-частотные” данные, и такое представление, определяемое как Гильбертов спектр. $r_j(t) = r_{j-1} - c_j(t)$

Допустим, что имеется произвольный сигнал $y(t)$. Сущность метода EMD заключается в последовательном вычислении функций эмпирических мод $c(t)$ и остатков $r_j(t) = r_{j-1} - c_j(t)$, где $j = 1,2,3,..,n$ при $r_0 = y(t)$. Результатом разложения будет представление сигнала в виде суммы модовых функций и конечного остатка:

$$x(t) = \sum_{j=1}^{n} c_j(t) + r_n(t), \qquad (1.15)$$

где n — количество эмпирических мод, которое устанавливается в ходе вычислений.

Алгоритм эмпирической декомпозиции сигнала складывается из

следующих операций его преобразования:

1. Находим в сигнале $y(k)$ положение всех локальных экстремумов, максимумов и минимумов процесса (номера точек $k_{i.ext}$ экстремумов), и значения в этих точках (рис. 8). Между этими экстремумами сосредоточена вся информация сигнала. Группируем раздельно для максимумов и для минимумов массивы координат $k_{i.ext}$ и соответствующих им амплитудных значений $y(k_{i.ext})$. Число строк в массивах максимумов и минимумов не должно отличаться более чем на 1.

Рисунок 8 – Локализация экстремумов в сигнале

2. Применяя сплайны (или каким либо другим методом) вычисляем верхнюю $u_t(k)$ и нижнюю $u_b(k)$ огибающие процесса соответственно, по максимумам и минимумам, как это показано на рис. 9. Определяем функцию средних значений $m_1(k)$ между огибающими (рис. 9).

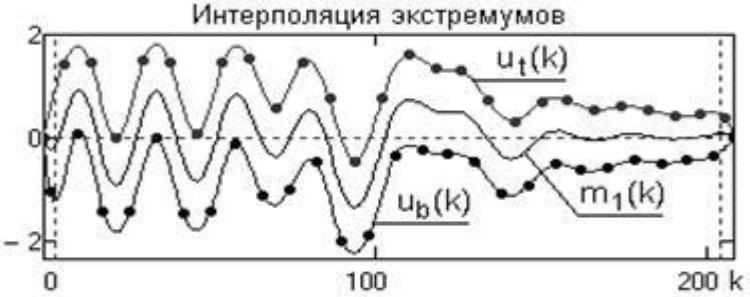

Рисунок 9 – Интерполяция экстремумов и построение огибающих

Определяем функцию средних значений $m_1(k)$ между огибающими.

$$m_1(k) = \frac{(u_t(k) + u_b(k))}{2}, \quad (1.16)$$

Разность между сигналом $y(k)$ и функцией $m_1(k)$ даёт нам первую компоненту отсеивания –функцию $h_1(k)$, которая является первым приближением к первой функции IMF:

$$h_1(k) = y(k) - m_1(k), \quad (1.17)$$

3. Повторяем операции 1 и 2, принимая вместо $y(k)$ функцию $h_1(k)$, и находим второе приближение к первой функции IMF – функцию $h_2(k)$.

$$h_2(k) = h_1(k) - m_2(k), \quad (1.18)$$

Последующие итерации выполняются аналогичным образом:

$$h_i(k) = h_{i-1}(k) - m_i(k), \quad (1.19)$$

По мере увеличения количества итераций функция $m_i(k)$ стремится к нулевому значению, а функция $h_i(k)$ – к неизменяемой форме.

Последнее значение $h_i(k)$ итераций принимается за наиболее высокочастотную функцию $r_1(k) = y(k) - c_1(k)$ семейства IMF, которая непосредственно входит в состав исходного сигнала $y(k)$. Это позволяет вычесть $c_1(k)$ из состава сигнала и оставить в нем более низкочастотные составляющие

$$r_1(k) = y(k) - c_1(k) \quad , \tag{1.20}$$

На рис. 10 показано графическое представление вычитание из сигнала высокочастотной составляющей, сформированной по алгоритму, заданному (1.15)-(1.19).

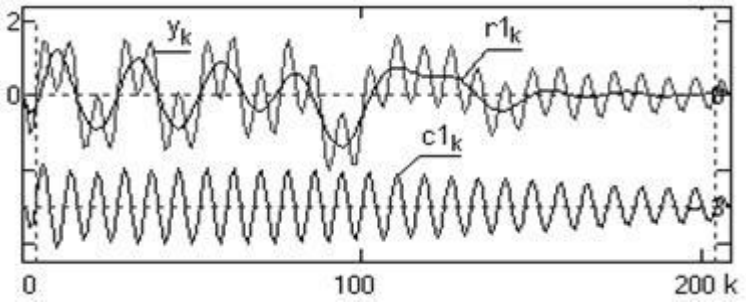

Рисунок 10 – Выявление низкочастотных составляющих в сигнале

Следующую внутреннюю модовую функцию найдем, повторив операции 1-3 декомпозиции, описанные выше, с той разницей, что входным сигналом является остаток $r_1(k)$.

Шаги 1-3 могут быть повторены для всех последовательных $r_j(k)$, и результат будет представлять последовательность вычислений:

$$r_1(k) - c_2(k) = r_2(k), \ldots, r_{n-1}(k) - c_n(k) = r_n(k), \tag{1.21}$$

Метод EMD закончен, когда остаток, в идеале, не содержит экстремумов. Это означает, что остаток – или константа или монотонная функция. Извлечённые IMFs симметричны, имеют уникальные локальные частоты, различные IMFs не показывают ту же самую частоту в то же самое время. Другими словами, остановка декомпозиции сигнала должна происходить при максимальном «выпрямлении» остатка, т.е. превращения

его в тренд сигнала по интервалу задания с числом экстремумов не более 2-3 [17].

1.3.4 Выводы

Преобразование Фурье и вейвлет-преобразование заслуженно получили широкую известность благодаря использованию в них хорошо обоснованных математических методов и наличию эффективных алгоритмов их реализации. Кроме того, оба это преобразования, как показала практика, достаточно универсальны и могут с успехом применяться в различных областях. Но для практического использования хотелось бы иметь преобразование, не только позволяющее работать с нестационарными процессами, но и использующее адаптивный, определяемый исходными данными базис преобразования. Поэтому для данной задачи выбран метод Гильберта-Хуанга, так как он удовлетворяет всем необходимым условиям.

2 Проектирование модуля голосовой идентификации диктора

Использование модуля голосовой идентификации диктора в системе распознавания речи предполагает увеличения коэффициента качества распознавания, а также повышение дикторонезависимости системы.

2.1 Структурная схема

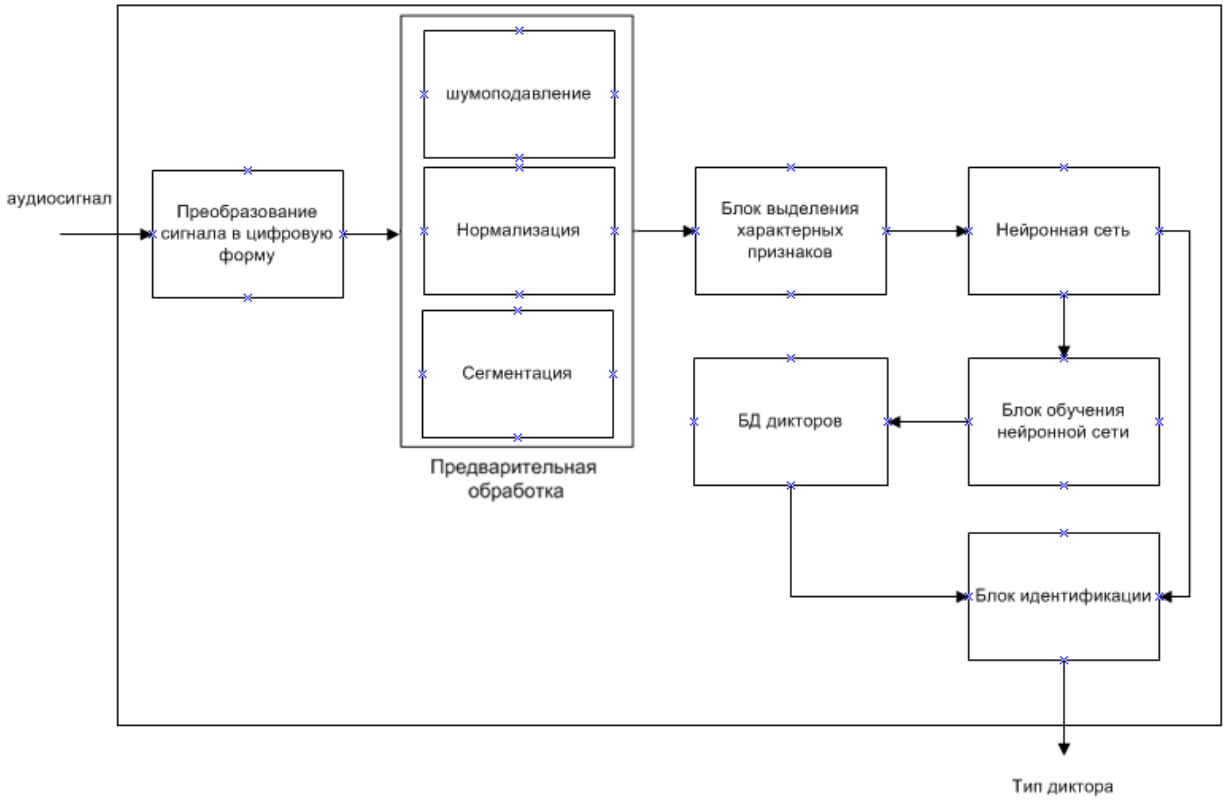

Рисунок 11 - Структурная схема модуля ГИД в системе распознавания речи

На вход модуля голосовой идентификации диктора поступает аудиосигнал запись которого происходит через микрофон, подключённый к входу звуковой карты компьютера, или используется уже ранее записанный сигнал. Акустический компонент преобразует сигнал в цифровую форму с заданными параметрами частоты дискретизации. Затем осуществляется

очистка сигнала от шума, удаление не несущих информации участков, проводится нормализация сигнала и его разбиение на фиксированные интервалы во временной области, на которых будут определяться характеристики. На блоке выделение характерных признаков происходит выделение характерных признаков сигнала с помощью преобразования Гильберта-Хуанга. В блок нейронная сеть попадает результат с предыдущего блока, дальнейшие действия зависят от выбора режима работы. При выборе «обучение» данные передаются на блок обучения нейронной сети, где происходит её обучение и передача в базу данных. При выборе режима «распознавание» данные переходят на блок идентификации, так же в этот блок приходят данные с блока база данных с уже сохранёнными данными. Далее происходит идентификация и классификация (определение принадлежности к определённому классу), затем следует отправление результата типа диктора.

2.2 Алгоритм работы модуля

Для начала работы подаётся аудиосигнал, затем происходит выбор режима работы модуля. При выборе режима «Обучение», первым делом, происходит преобразование Гильберта-Хуанга, после преобразованный сигнал подаётся на нейронную сеть, где происходит обучение нейронной сети на характерные признаки голоса диктора и сохранение её. При выборе режима «Распознавание» происходит загрузка обученной нейронной сети и выполняется преобразование Гильберта-Хуанга. Преобразованный сигнал подаётся на нейронную сеть для сравнение с характерными признаками голоса диктора. Далее происходит выбор класса диктора, по принципу максимально схожих характерных признаков и выводиться результат.

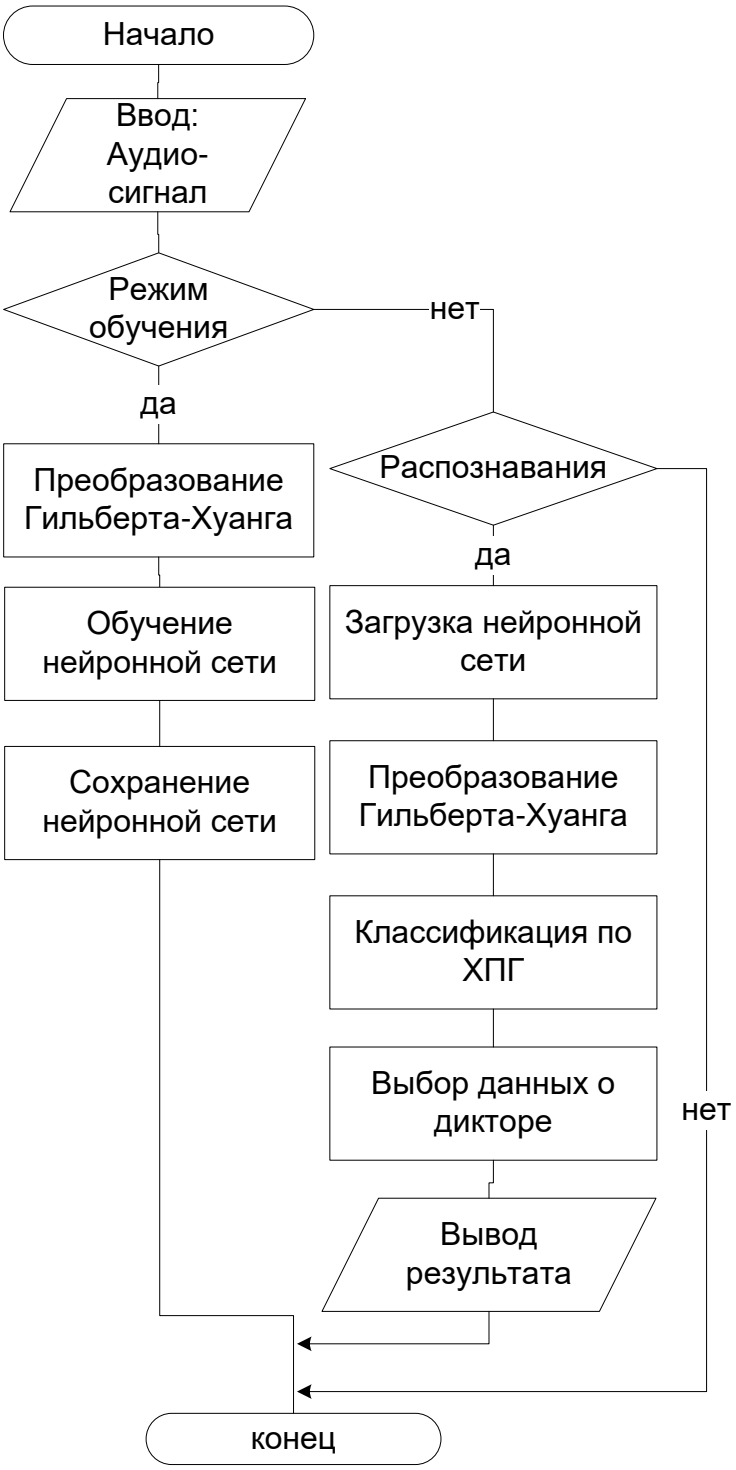

Рисунок 12 – Алгоритм работы модуля

2.3 Методы классификации речевого сигнала

В данном подразделе рассматриваются основные существующие решения задачи идентификации диктора по голосу. Несмотря на то, что методы во многом отличаются, в целом можно выделить следующие основные этапы, присущие каждому из рассматриваемых методов:

1. Извлечение признаков из входного речевого сигнала.
2. Построение модели (шаблона) диктора на основе полученных на предыдущем шаге векторов признаков.

Процесс определения диктора, зарегистрированного в системе, по входному речевому сигналу во всех рассматриваемых методах состоит в поиске наиболее подходящей сохранённой модели на основе каких-либо критериев [10].

2.3.1 Dynamic Time Warping

Dynamic Time Warping (DTW) – метод динамического программирования, позволяющий найти близость между двумя последовательностями измерений за некоторый промежуток времени. В общем случае эти последовательности могут быть разной длины, и измерения могут производиться с разной скоростью.

В качестве сохраняемой модели в данном методе выступает последовательность векторов признаков входного речевого сигнала из обучающей выборки $Q = \{q_1, \dots, q_n\}$. Пусть $C = \{c_1, \dots, c_m\}$ – последовательность векторов признаков входного речевого сигнала из тестовой выборки. Также вводятся понятия матрицы выравнивания двух последовательностей $M_{m \times n}$, в позиции (i, j) которой содержится значение выравнивания между элементами c_i и q_j последовательностей C и Q соответственно, и набора индексов смежных элементов этой матрицы $W = \{w_1, \dots, w_T\}$, определяющего соответствие между элементами сопоставляемых последовательностей. При этом элементы набора W должны удовлетворять следующим условиям:

1. $w_1 = (1,1),\ w_T = (m,n)$
2. Если $w_{t-1} = (a', b')$, то $w_t = (a, b)$, где $a - a' \leq 1,\ b - b' \leq 1$, (2.1)

Целью алгоритма DTW является нахождение такого набора W, удовлетворяющего условиям 1 и 2, при котором суммарное искажение последовательности C относительно последовательности Q было бы минимальным, то есть:

$$DTW(Q,C) = \min\left\{\frac{1}{T}\sqrt{\sum_{t=1}^{T} M(w_t)}\right\}, \quad (2.2)$$

Значение этого выражения и будет определять меру близости последовательностей Q и C. Для нахождения значения DTW(Q, C) применяется метод динамического программирования, где на каждом шаге вычисляется значение M(i, j) по формуле:

$$M(i,j) = d(i,j) + \min\{M(i-1,j-1), M(i-1,j), M(i,j-1)\} \quad (2.3)$$

При этом M(0, 0)=0, а все остальные элементы столбца и строки с индексом 0 инициализируются значением ∞. d(i, j) определяет евклидово расстояние между элементами c_i и q_j. Результатом работы алгоритма является значение DTW(Q, C) = M(m, n).

Для определения диктора вычисляется значение $DTW(Q_i, C)$, для каждого сохраненного шаблона. Значение i, при котором достигается минимум, определяет номер диктора, образец голоса которого наиболее близок к образцу входного речевого сигнала [15].

Основным преимуществом алгоритма DTW является простота реализации. Тем не менее, данный алгоритм неприменим для решения задачи текстонезависимой идентификации диктора.

2.3.2 Hidden Markov Model

Hidden Markov Model (HMM) – статистическая модель, которая может использоваться для решения задачи классификации скрытых параметров на основе наблюдаемых. HMM представляет собой конечный автомат, в котором переходы между состояниями осуществляются с некоторой вероятностью, и задано стартовое состояние, с которого начинается процесс. Через дискретные моменты времени может осуществляться переход в новые состояния. При этом каждому скрытому состоянию с заданной вероятностью соответствует наблюдаемое состояние. Кроме того, текущее состояние автомата зависит только от конечного числа предыдущих, а закон смены состояний не меняется во времени [15].

2.3.3 Vector Quantization

Задача векторного квантования с кодовыми векторами $W = \{w_1, w_2, \dots, w_n\}$ для последовательности входных векторов C = $\{c_1, c_2, \dots, c_m\}$ ставится как задача минимизации искажения при замещении каждого вектора из Q соответствующим кодовым вектором. Моделью диктора в данном методе является множество кодовых векторов, получаемое из входной последовательности векторов признаков речевого сигнала. Для построения этого множества исходная последовательность векторов признаков разбивается на L кластеров, и в качестве кодовых векторов берутся их центры.

Процесс определения диктора по входному речевому сигналу происходит следующим образом. Для каждого тестового вектора c_i определяются k ближайших кодовых векторов. Пусть k_{ij} – количество векторов, принадлежащих диктору S_j,среди найденных ближайших. Тогда вероятность того, что вектор c_i принадлежит диктору S_j, определяется формулой:

$$P(S_j|c_i) = \frac{k_{ij}}{k} \quad , \tag{2.4}$$

Таким образом, последовательность тестовых векторов может быть классифицирована по правилу:

$$S = \underset{1 \le j \le N}{\operatorname{argmax}} \prod_{i=1}^{L} P(S_j|c_i) \quad , \tag{2.5}$$

Для сглаживания погрешности измерения, связанной с близкими к нулю вероятностями, с учётом постоянности числа чаще используют правило:

$$S = \underset{1 \le j \le N}{\operatorname{argmax}} \sum_{i=1}^{L} P(S_j|c_i) = \underset{1 \le j \le N}{\operatorname{argmax}} \sum_{i=1}^{L} k_{ij} \quad , \qquad (2.6)$$

Метод векторного квантования прост в реализации, применим к задаче текстонезависимой идентификации диктора, однако не всегда дает высокую точность распознавания [19].

2.3.4 Support Vector Machine

Support vector machine (метод опорных векторов) – бинарный классификатор, который строит в пространстве признаков разделяющую функцию, задающую гиперплоскость, вида:

$$f(x) = w \cdot x + b \text{ ,} \tag{2.7}$$

Пусть задана последовательность точек пространства признаков X = $\{x_1, x_2, \dots, x_n\}$ с метками $Y = \{y_1, y_2, \dots y_n\},\ y_i \in \{-1,1\},\ 1 \le i \le n,$ соответствующими двум классам. В случае линейной разделимости данных условия для нахождения функции f(x) записываются в виде:

$$\begin{cases} w \cdot x_i + b \ge 1, & y_i = 1 \\ w \cdot x_i + b \le -1, & y_i = -1 \end{cases} \Leftrightarrow y_i(w \cdot x_i + b) - 1 \ge 0, 1 \le i \le n \tag{2.8}$$

Для надежного разделения классов необходимо чтобы расстояние между разделяющими гиперплоскостями было как можно большим. Расстояние вычисляется как $\frac{2}{\|w\|}$, следовательно, задачу поиска разделяющей гиперплоскости можно свести к минимизации ‖w ‖ при указанных условиях. Эта задача может быть решена с помощью метода множителей Лагранжа [6].

В случае линейно-неразделимых множеств вводится функция ядра. Основная идея заключается в том, чтобы отобразить исходное пространство в пространство более высокой размерности, в котором множества уже могут быть разделимы линейно. При этом в силу того что всюду в алгоритме признаки используются не отдельно, а в виде скалярных произведений, нет необходимости строить данное преобразование в явном виде. Достаточно

задать функцию ядра, определяющую скалярное произведение в новом пространстве:

$$K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j). \quad (2.9)$$

В качестве сохраняемой модели диктора в методе опорных векторов выступают параметры разделяющей функции f(x) , а также параметры функции ядра. Параметры ядра обычно определяют путем перебора некоторого множества значений и оценкой методом кросс-валидации.

После того, как решающая функция f(x) вычислена, принадлежность вектора x' соответствующему классу определяется знаком выражения $f(x')$.

Для применения метода опорных векторов к задаче многоклассового распознавания используется стратегия «один против остальных». Для этого строятся q классификаторов, каждый из которых обучается отличать один конкретный класс от всех остальных. При распознавании объект приписывается к тому классу, чей классификатор выдал наибольшее значение разделяющей функции f(x) [19].

Метод опорных векторов дает высокую точность классификации, имеет теоретическое обоснование, позволяет применять различные подходы к классификации в соответствии с выбором функции ядра. Среди недостатков следует отметить проблему выбора ядра, а также медленное обучение в случае задачи многоклассового распознавания.

2.3.5 Gaussian Mixture Model

Модель гауссовых смесей представляет собой взвешенную сумму M компонент и может быть описана выражением:

$$P(\bar{x}|\lambda) = \sum_{i=1}^{M} p_i b_i(\bar{x}), \tag{2.10}$$

где $\bar{x}$ – D-мерный вектор случайных величин, $p_i, 1 \le i \le M$ – веса компонентов модели, $b_i(\bar{x}), 1 \le i \le M$ – функции плотности распределения составляющих модели:

$$b_i(\bar{x}) = \frac{1}{(2\pi)^{D/2} |\sum_i|^{1/2}} exp\left\{-\frac{1}{2}(\bar{x} - \bar{\mu}_i)^T \sum_i^{-1} (\bar{x} - \bar{\mu}_i)\right\}, \tag{2.11}$$

где $\bar{\mu}_i$ – вектор математического ожидания и Σi – ковариационная матрица. При этом веса смеси удовлетворяют условию:

$$\sum_{i=1}^{M} p_i = 1 \quad , \tag{2.12}$$

Полностью модель гауссовой смеси определяется векторами математического ожидания, ковариационными матрицами и весами смесей для каждого компонента модели:

$$\lambda = \left\{p_i, \bar{\mu}_i, \sum_i\right\}, \ i = 1, \dots M \quad , \tag{2.13}$$

При использовании данного метода каждый диктор представляется моделью гауссовых смесей λ.

Для построения модели диктора необходимо оценить её параметры, которые наилучшим образом соответствуют распределению векторов признаков обучающего высказывания. Наиболее популярным и широко используемым методом решения этой задачи является метод оценки максимального правдоподобия. Целью оценки максимального правдоподобия является нахождение параметров модели, которые максимизируют правдоподобие этой модели, при заданных обучающих данных.

Для последовательности обучающих векторов $X = \{\overline{x_1}, \ \dots, \ \overline{x_T}\}$ правдоподобие модели гауссовых смесей может быть записано в виде:

$$P(X|\lambda) = \prod_{t=1}^{T} P(\overline{x_t}|\lambda) \ , \qquad (2.14)$$

Это выражение представляет нелинейную функцию от параметров λ, и ее непосредственное вычисление невозможно, поэтому обычно для оценки параметров применяется EM-алгоритм [16].

Пусть $S = \{S_1, S_2, \dots, S_N\}$ – группа дикторов, которые представлены набором моделей гауссовых смесей $\lambda_1, \ \lambda_2, \ \dots, \ \lambda_N$.

При идентификации диктора требуется найти модель, которая имеет наибольшее значение апостериорной вероятности для заданного высказывания:

$$S = \underset{1 \le k \le N}{\operatorname{argmax}} P(\lambda_k|X) = \underset{1 \le k \le N}{\operatorname{argmax}} \frac{P(X|\lambda_k)P(\lambda_k)}{P(X)} = \underset{1 \le k \le N}{\operatorname{argmax}} P(X|\lambda_k) \ , \qquad (2.15)$$

Используя логарифм и независимость между наблюдениями, система идентификации диктора в итоге вычисляет:

$$S = \underset{1 \le k \le N}{\operatorname{argmax}} \sum_{t=1}^{T} \log P(\overline{x_t} | \lambda_k) \text{,} \tag{2.16}$$

Модели гауссовых смесей представляют собой эффективный алгоритм с высокой точностью распознавания. Вместе с тем возникает ряд проблем, связанных с выбором числа компонентов модели и инициализацией её начальных параметров [19].

2.3.6 Нейронная сеть

В любой системе распознавания речи всегда присутствует этап сравнения входного сигнала с имеющимися эталонами. Вне зависимости от наличия или отсутствия предварительной обработки сигнала (выделение основных признаков, преобразование в другую форму в новом параметрическом пространстве и т. д.) сигнал представляет собой вектор в установленном параметрическом пространстве, который в дальнейшем будет сравниваться с векторами, хранящимися в памяти для определения его принадлежности к определённому классу. Такая классификация образов является одной из основных задач решаемых с помощью нейронной сети,

среди которых наиболее распространены:

- Распознавание зрительных, слуховых образов; огромная область применения: от распознавания текста и целей на экране радара до систем голосового управления;
- ассоциативный поиск информации и создание ассоциативных моделей;
- формирование моделей различных нелинейных и трудно описываемых математически систем, прогнозирование развития этих систем во времени; применение на производстве; прогнозирование природных процессов, изменений курсов и т.д;
- системы управления и регулирования с предсказанием; управление роботами, другими сложными устройствами - разнообразные конечные автоматы: системы массового обслуживания и коммутации, телекоммуникационные системы;
- принятие решений и диагностика, исключающие логический вывод; особенно в областях, где отсутствуют чёткие математические модели: в медицине, криминалистике, финансовой сфере;

Выделим характерные свойства искусственных нейронных сетей:

- Обучаемость. Одним из этапов функционирования нейронной сети является обучение, в процессе которого на ее вход поочерёдно поступают данные из обучающего набора с целью корректировки весовых коэффициентов синаптических связей для получения наиболее адекватного сигнала на выходе нейронной сети.
- Способность к обобщению. Отклик сети после обучения может быть до некоторой степени нечувствителен к небольшим изменениям входных сигналов (шуму или вариациям входных образов).
- Способность к абстрагированию. Если при обучении предъявить сети несколько искажённых вариантов входного образа, то сеть может создать на выходе идеальный образ, который не входил в обучение.
- Параллельность обработки и реализуемости нейронных сетей.
- Универсальность. Хотя почти для всех перечисленных задач существуют эффективные математические методы решения и, несмотря на то, что сети проигрывают специализированным методам; благодаря универсальности и перспективности они являются важным направлением исследования, требующим тщательного изучения [13].

Модель искусственного нейрона. Искусственные нейронные сети индуцированы биологией, так как они состоят из элементов, функциональные возможности которых аналогичны большинству элементарных функций биологического нейрона. На рисунке 13 показана обобщённая модель нейрона, используемая в качестве основного строительного блока в нейронных сетях.

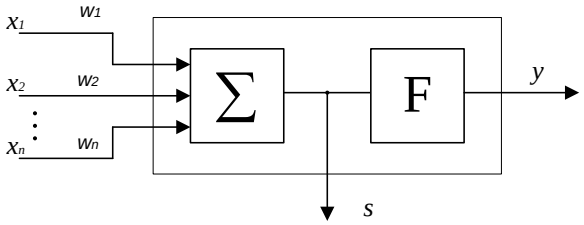

Рисунок 13 – Искусственный нейрон с активационной функцией

На входы нейрона подаётся множество сигналов, каждый из которых умножается на вес, и затем произведения складываются. Результат суммирования s (2.17) служит аргументом функции активации. Значение функции активации соответствует отклику нейрона y (2.18) на произвольную комбинацию входных воздействий. Другими словами, посредством активации нейрона осуществляется трансформация множества входных воздействий в выходной сигнал с желаемыми характеристиками.

$$s = \sum_{i=1}^{p} w_i x_i + w_0 \,, \tag{2.17}$$

$$y = f(s) \,, \tag{2.18}$$

где w_i - вес синапса, ($i=0,1,2...p$);

w_0 - значение смещения;

s - результат суммирования;

x_i - компонента входного вектора (входной сигнал), ($i=1,2,...p$);

y - выходной сигнал нейрона;

p - число входов нейрона;

f - нелинейное преобразование (функция активации).

В общем случае входной сигнал, весовые коэффициенты и значения

смещения могут принимать действительные значения. Выход *y* определяется видом функции активации и может быть как действительным, так и целым. Во многих практических задачах входы, веса и смещения могут принимать лишь некоторые фиксированные значения. Синаптические связи с положительными весами называют возбуждающими, с отрицательными весами - тормозящими. Таким образом, нейрон полностью описывается своими весами w_i и передаточной функцией *f(x)*. Вместе с правилами корректировки весовых коэффициентов на входе нейрона, правилами обучения, отличительной особенностью многих нейронных структур является выбор функции активации *f*. Активационная функция может быть обычной линейной функцией (2.19).

$$y=k(s), \tag{2.19}$$

где *k* – постоянная пороговой функции;

$$y = \begin{cases} 0, s \le T \\ 1, s > T \end{cases}, \tag{2.20}$$

где T – некоторая постоянная пороговая величина.

Если функция активации сужает диапазон изменения величины *s* (2.17) так, что при любых значениях *s* значения *y* (2.18) принадлежат некоторому конечному интервалу, то *f* называется “сжимающей” функцией. В качестве “сжимающей” функции часто используется логистическая или “сигмоидальная” (S-образная) функция, показанная на рисунке 14.

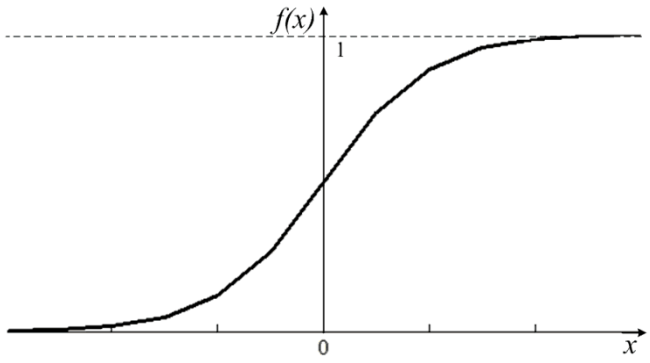

Рисунок 14 - Сигмоидальная функция

Эта функция задаётся формулой (2.21).

$$f(x) = \frac{1}{1 + \exp(-x)}\,. \tag{2.21}$$

Другой широко используемой активационной функцией является гиперболический тангенс (2.22).

$$f(x) = th(x)\,. \tag{2.22}$$

График функции гиперболического тангенса представлен на рисунке 15.

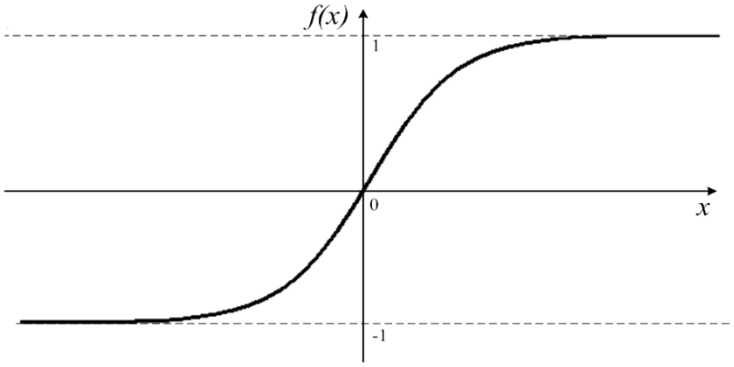

Рисунок 15 – функция гиперболического тангенса

Хотя один нейрон и способен выполнять простейшие процедуры распознавания, сила нейронных вычислений проистекает от соединений нейронов в сетях.

Нейронная сеть представляет собой структуру нейронов, соединённых между собой и характеризуется их внутренними свойствами, индивидуальной топологией (архитектурой), а также правилами обучения для получения желаемого выходного сигнала.

Сеть обучается, чтобы для некоторого множества входов давать желаемое множество выходов. Каждое такое входное (или выходное) множество рассматривается как вектор. Обучение осуществляется путём последовательного предъявления входных векторов с одновременной подстройкой весов в соответствии с определённой процедурой. В процессе обучения веса сети постепенно становятся такими, чтобы каждый входной вектор вырабатывал выходной вектор.

Существуют три алгоритма обучения: "с учителем", "без учителя"

(самообучение) и смешанное. В первом случае нейронная сеть располагает правильными ответами (выходами сети) на каждый входной пример. Веса настраиваются так, чтобы сеть производила ответы как можно более близкие к известным. Обучение без учителя не требует знания правильных ответов на каждый пример обучающей выборки. В этом случае раскрывается внутренняя структура данных или корреляции между образцами в системе данных, что позволяет распределить образцы по категориям. При смешанном обучении часть весов определяется посредством обучения с учителем, в то время как остальная получается с помощью самообучения.

Рассмотрим ряд основных архитектур нейронных сетей, успешно применяемых для решения задачи классификации, одна из постановок которой представлена в данном дипломном проекте [13].

Персептрон. Наиболее простой нейросетевой архитектурой является однослойный персептрон. Он состоит из одного слоя искусственных нейронов, соединённых с помощью весовых коэффициентов со множеством входов (рисунок 16). Слой - это совокупность нейронов с единым входным сигналом. Элемент **Σ** умножает каждый вход x на вес w и суммирует взвешенные входы. Если эта сумма больше заданного порогового значения, выход равен единице, в противном случае – нулю.

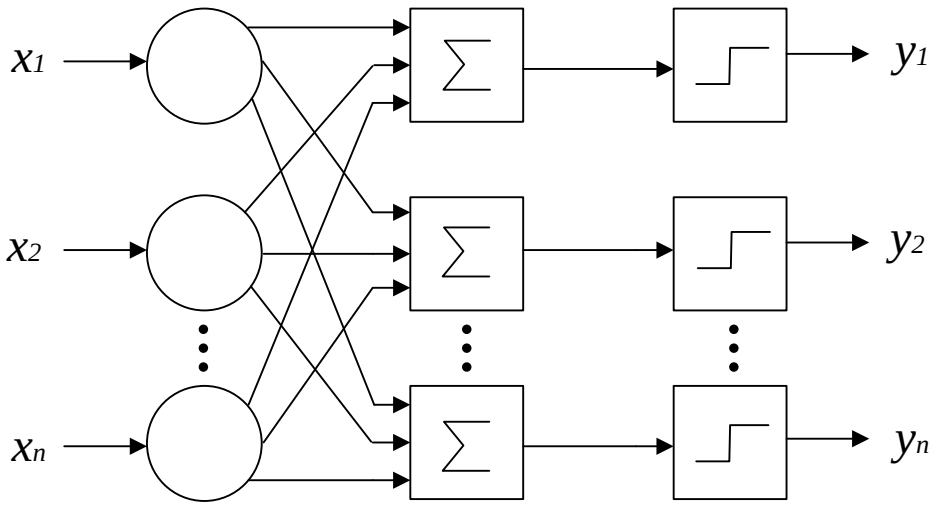

Рисунок 16 – Персептрон

Персептрон обучают, подавая множество образов по одному на его вход и подстраивая веса до тех пор, пока для всех образов не будет достигнут требуемый выход.

Применение такой топологии оправдано только для задач, обладающих высокой линейностью. Например, можно построить нейронную сеть, разбивающую точки (0,0) и (1,1) на два класса для двумерного сигнала, но невозможно решить задачу по разбиению точек (0,0), (1,1) – в первый класс, и (0,1), (1,0) - во второй. Это широко известный пример неспособности

простого персептрона решить задачу «исключающее или».

Имеется обширный класс функций, не реализуемых однослойной сетью. Вероятность того, что случайно выбранная функция окажется линейно разделимой, весьма мала даже для умеренного числа переменных. По этой причине однослойные персептроны на практике ограничены простыми задачами [13].

Многослойный персептрон. Серьёзное ограничение однослойными сетями можно преодолеть, добавив дополнительные слои. Например, двухслойные сети можно получить каскадным соединением двух однослойных сетей. Они способны выполнять более общие классификации. Такая сеть может моделировать функцию практически любой степени сложности, причём число слоёв и число элементов в каждом слое определяют сложность функции.

На рисунке 17 изображена двухслойная сеть, которая может обучаться с помощью процедуры обратного распространения.

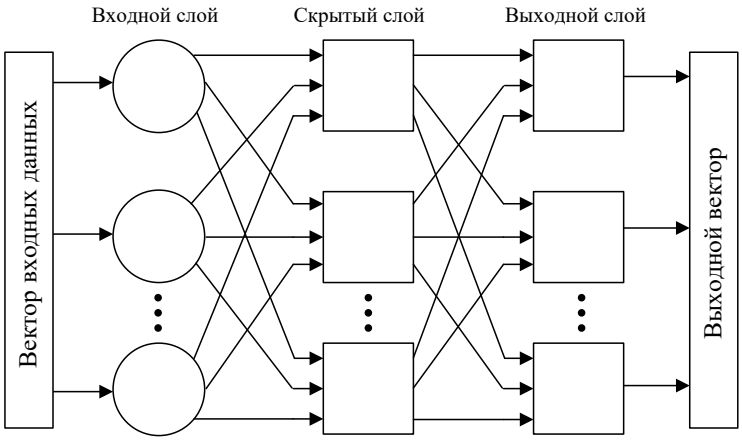

Рисунок 17 – Двухслойный персептрон

Нейроны одного и того же слоя друг с другом не связаны, и каждый нейрон связан со всеми нейронами последующего слоя (кроме последнего слоя - его выходы являются выходами сети в целом). Первый слой нейронов

(соединённый с входами) служит лишь в качестве распределительных точек, суммирования входов здесь не производится. Входной сигнал просто проходит через них к весам на их выходах. Получив входные сигналы, нейрон следующего слоя суммирует их с соответствующими весами, затем применяет к этой сумме передаточную функцию и передаёт результат на один из входов нейрона второго слоя, который в свою очередь, суммирует полученные от первого слоя сигналы с некоторыми весами и т.д. Прямое функционирование сети описывается следующими соотношениями:

$$y^0 = x, \tag{2.23}$$

$$y_j^k = f\left(\sum_{i=1}^{n(k-1)} w_{ij}^k y_i^{k-1}\right),\ j = 1 : n(k), \tag{2.24}$$

где x - входной сигнал;

y_j^k - значение j-го выхода нейрона k-го слоя;

w_{ij}^k - вес связи от i -го нейрона $(k-1)$-го слоя к j-му нейрону k-го слоя;

f - функция активации;

$n(k)$ - число нейронов в k-м слое.

В качестве активационной функцией в сетях обратного распространения обычно используется сигмоидальная функция. Многослойные нейронные сети обладают большей представляющей мощностью, чем однослойные, только в случае присутствия нелинейности. Сжимающая функция обеспечивает требуемую нелинейность [13].

Для обучения многослойных нейронных сетей применяется алгоритм обратного распространения ошибки. Если при прямом функционировании входной сигнал распространяется по сети от входного слоя к выходному, то при подстройке весов ошибка сети распространяется от выходного слоя к входному.

2.3.7 Выводы

На данный момент существуют методы классификации речевого сигнала, позволяющих решать задачу идентификации диктора по голосу. Многие из их ещё находятся в стадии разработки и закрыты от общего обзора, поэтому решено разработать свой модуль идентификации диктора. Для решения задачи классификации рационально использовать нейронную сеть с архитектурой трёхслойного персептрона. К её достоинствам можно отнести сравнительную простоту анализа и достаточно высокую эффективность классификации. Благодаря использованию непрерывной функции возбуждения такие сети способны к обобщению обучающей выборки.

3 Программная реализация

Средой для разработки программы был выбран программный пакет MatLab R15b, т.к. в состав пакета входят готовые библиотеки для работы со звуковыми сигналами. Также в состав MatLab входит среда Guide Builder для создания приложений с графическим интерфейсом пользователя. Для моделирования структуры нейронной сети был выбран пакет Neural Networks. Wavelet Toolbox, входящий в состав MatLab является очень удобный инструментом для изучения и проведения вейвлет-преобразований. В стандартный пакет аудио поддержки системы Matlab включены функции, позволяющие произвести запись звукового сигнала.

3.1 Описание работы программы

Перед началом работы пользователь должен добавить БД и обучить нейронную сеть. Все эти операции осуществляются через графический интерфейс. Также есть возможность управления записью/воспроизведением звука, открытия и сохранения звуковых файлов с помощью диалоговых окон, а также графического отображения речевого сигнала.

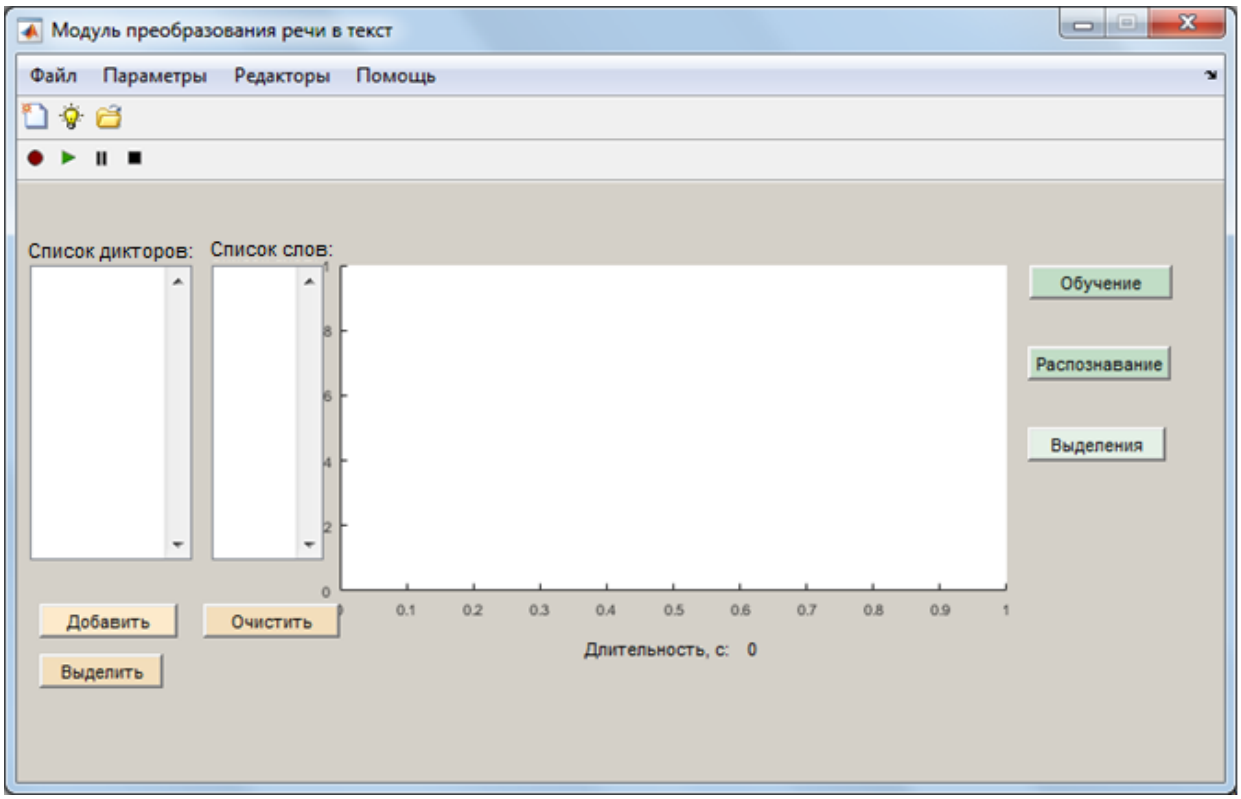

Рисунок 18 – Пользовательский интерфейс программы

На панели управления располагаются кнопки «Play», «Record», «Pause» и «Stop». При помощи которых можно записать и прослушать записанный сигнал. В графической области отображается записанный аудиосигнал или загруженный файл. При нажатии на кнопку «Режим выделения» графическая область поделиться на сектора, для более удобного выделения. Выделить нужный фрагмент из потока аудиосигнала, можно выделив нужный отрезок на графической области и нажать на кнопку «Добавить». Данный сегмент отобразится в списке входных сигналов.

Обучающая выборка формируется с помощью команды «Добавить», путём добавления нужных слов эталонов из «Списка сигналов» в «Список слов» с указанием содержания в диалоговом окне. При этом выполняется предварительная обработка, и рассчитываются признаки выбранных слов.

Сформированную обучающую выборку признаков слов можно сохранить на диск для последующего использования. Сохранение и загрузка выборки выполняется с помощью диалоговых окон, вызываемых через соответствующие команды в меню «Файл» (рисунок 19).

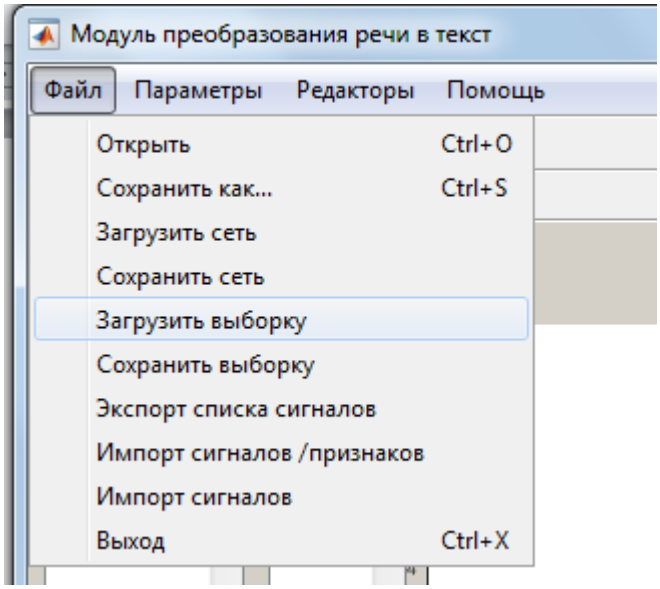

Рисунок 19 – Команды в меню «Файл»

Функция «Обучение» запускает процедуру обучения нейронной сети на сформированном обучающем множестве с заданными параметрами.

Обучение будет направленно на отношение характерных признаков голоса к одному из трёх классов:

- Первый класс – детский голос
- Второй класс – мужской голос
- Третий класс – женский голос

В дальнейшем можно будет увеличить количество классов для более лучшего результата.

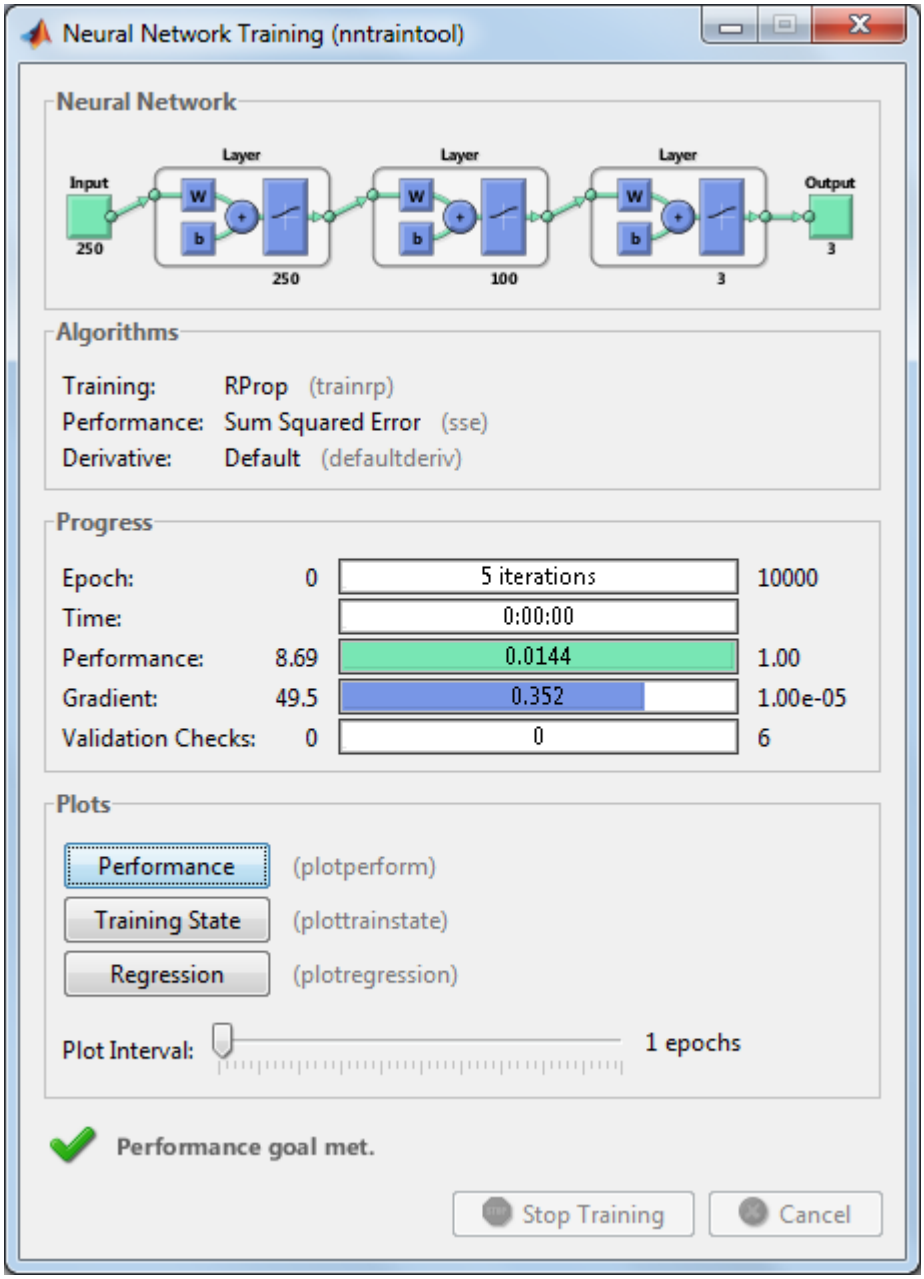

Рисунок 20 – Обучение нейронной сети

В «Neural Network» показано, что нейронная сеть состоит из трёх слоёв (трёхслойный персептрон). Первый слой состоит из 250 нейронов, второй слой из 100 нейронов, 3 слой это количество классов.

Предоставляется возможность сохранить значения весов связей нейронной сети, обученной под конкретного диктора, для дальнейшего использования при распознавании. Импорт и загрузка сети осуществляется

через меню «Файл» (рисунок 19).

Для распознавания необходимо записать аудиосигнал или открыть заранее записанный файл. При нажатии кнопки «Распознать» начнётся процесс распознавания. Загружается уже обученная нейронная сеть, совершается преобразование Гильберта-Хуанга и аудио-сигнал отправляется на нейронную сеть. Там сравниваются характерные признаки и в текстовом окне на главной форме отображается наименование распознанного диктора (рисунок 21).

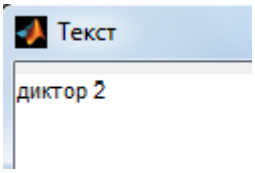

Рисунок 21 – распознанный диктор

3.2 Создание речевой базы для тестирования

Для наполнения речевой базы было привлечено 25 человек. Каждый диктор предоставил аудио-записи пяти ключевых фраз.

Также в тестирование были использованы записи, полученные с помощью синтеза речи. Для решения данной задачи были выбраны системы Google cloud Speech API и Yandex SpeechKit. тем самым, количество распознаваемых голосов в тесте выросло до 35 единиц.

SpeechKit Cloud API — это HTTP API, который позволяет разработчикам приложений использовать речевые технологии Яндекса:

- распознавание речи
- синтез речи[21].

SpeechKit, для наполнения голосовой базы, предоставил 4 женских и 2 мужских голоса (рисунок 22).

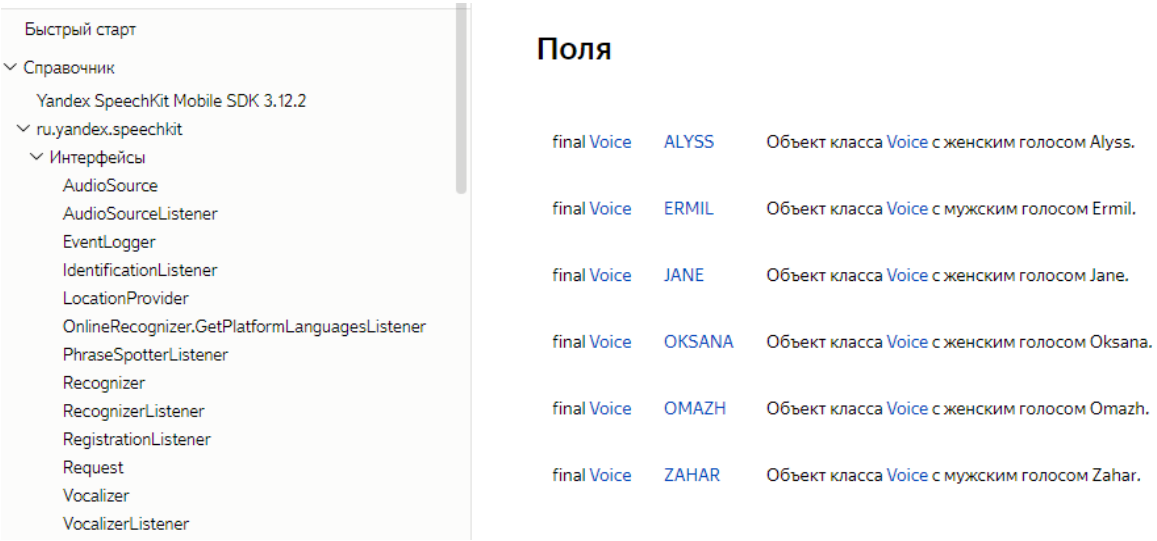

Рисунок 22 – список представленных голосов в SpeechKit

Speech API – интерфейс программирования приложений, основанный на технологии COM (технологический стандарт от компании Microsoft, предназначенный для создания программного обеспечения на основе взаимодействующих компонентов объекта, каждый из которых может использоваться во многих программах одновременно), предназначенный для распознавания и синтеза речи [20]. Speech API предоставил два женских и два мужских голоса, для наполнения голосовой базы.

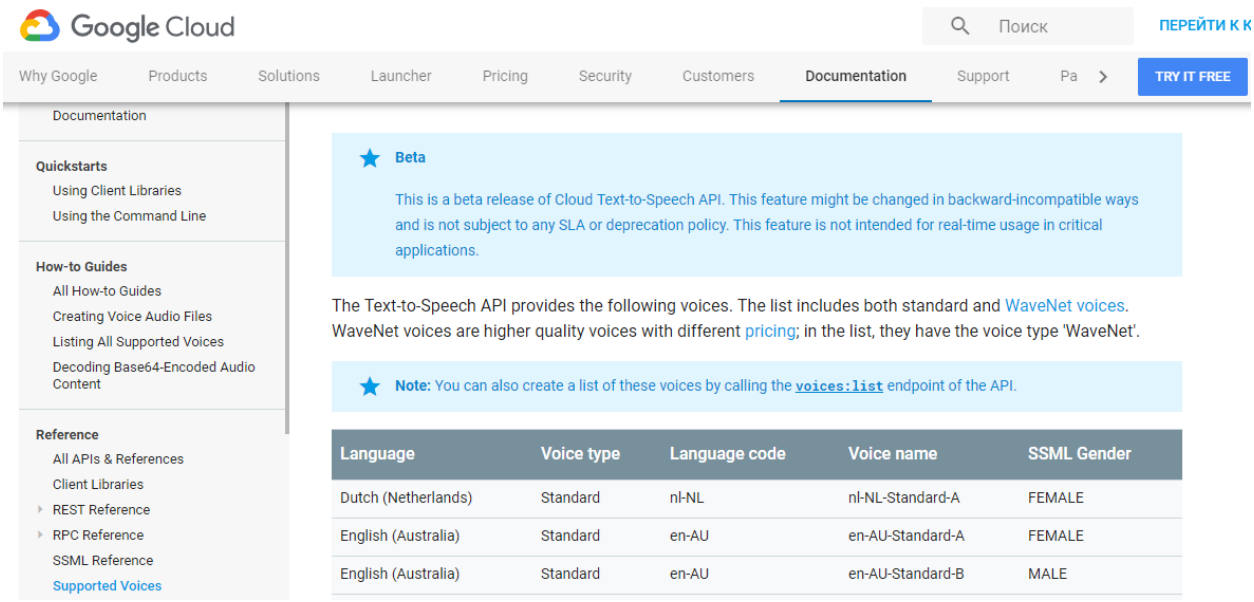

Рисунок 23 – Система Google cloud Speech API

3.3 Оценка качества работы модуля идентификации

Модуль голосовой идентификации диктора прошёл тестирование, для наблюдения различий было проведено сравнение с работой системы распознавания речи без модуля голосовой идентификации диктора (таблица 1).

Таблица 1 – Оценка качества работы модуля в системе распознавания речи

количество дикторов	Количество тестов	Коэффициент распознавания без модуля идентификации, %	Коэффициент распознавания с модулем идентификации, %
2	20	95	100
10	100	78	97
20	200	65	92
35	350	60.15	89.27

Были проведены тесты с разным количеством задействованных дикторов, относящихся к разным классам.

- Первый класс – мужской голос.
- Второй класс – женский голос.
- Третий класс – детский голос.

Из проведённых тестов видно, что распознавание диктора с помощью данного модуля в составе системы распознавания речи, повысило процент правильной идентификации диктора.

Модуль голосовой идентификации прошёл тест вне системы распознавания речи. Целью теста было определить качество классификации дикторов (таблица 2).

Таблица 2 – Оценка качества работы модуля

количество дикторов	Количество тестов	Качество классификации модуля, %
2	10	100
10	100	100
20	200	96.9
35	400	93.8

В таблице показаны данные тестов, проведённых с разным количеством дикторов, относящихся к разным классам.

- Первый класс – мужской голос.
- Второй класс – женский голос.
- Третий класс – детский голос.

Из проведённых тестов можно отметить, что модуль голосовой идентификации с высокой точностью определяет класс, к которому относится диктор.

3.4 Вывод

Данный модуль предназначен для улучшения работы системы распознавания речи. С помощью данного модуля система сможет распознавать дикторов по их характерным признакам голоса, что позволит увеличить надёжность распознавания диктора. Тесты показали, что надёжность распознавания диктора с модулем голосовой идентификации выросла на 20-30%.

ЗАКЛЮЧЕНИЕ

Были проанализированы существующие подходы и выбран алгоритм классификации пользователей по их голосовым характеристикам. Разработан метод выделения признаков речевого сигнала, позволяющий проводить идентификацию дикторов. Была выполнена программная реализация модуля голосовой идентификации диктора с использованием среды Matlab R15b. Определено повышение качества распознавания речи за счёт выбора оптимального классификатора, обученного на речевом материале, с схожими с диктором голосовыми характеристиками. В результате экспериментальных исследований разработанного модуля, отмечено увеличение надёжности распознавания на 20-30%.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Центр речевых технологий [Электронный ресурс] – режим доступа: http://www.speechpro.ru/.

2. Матвеев, Ю. Н. Технологии биометрической идентификации личности по голосу и другим модальностям // Вестник МГТУ им. Н. Э. Баумана. Электронное научно-техническое издание. 2012. № 3(3) [Электронный ресурс] – режим доступа: http://vestnik.bmstu.ru/catalog/it/biometric/91.html/

3. ОБЗОР ОСНОВНЫХ МЕТОДОВ РАСПОЗНАВАНИЯ ДИКТОРОВ Е. А. Первушин [Электронный ресурс] – режим доступа: http://cyberleninka.ru/article/n/obzor-osnovnyh-metodov-raspoznavaniya-diktorov.pdf.

4. Martin, A. Przybocki. The NIST 1999 Speaker Recognition Evaluation – An Overview // A. Martin. Digital Signal Processing. 2000. V. 10

5. Коваль, С. Л. Комплексная методика идентификации дикторов по голосу и речи // С. Л. Коваль. Информатизация и информационная безопасность правоохранительных органов: труды XX международной научной конференции. Москва.: Академия управления МВД России, 2011. С. 364-370.

6. Platt, J. C. Fast Training of Support Vector Machines using Sequential Minimal Optimization // J. C. Platt. Advances in Kernel Methods / Ed. by B. Scholkopf, C. C. Burges, A. J. Smola. MIT Press, 1999. P. 185–208.

7.Википедия [Электронный ресурс] – режим доступа: ru.wikipedia. /wiki/Преобразование_Фурье

8. Дьяконов, В. MATLAB: Учебный курс // В. Дьяконов. Санкт-Петербург.: Питер, 2001. - 560 с.

9. Распознавание речи. Часть 1. Классификация систем распознавания речи [Электронный ресурс] – режим доступа:https://geektimes.ru/post/64572/

10. Идентификация-диктора-по-голосу-текст [Электронный ресурс] – режим доступа: http://seminar.at.ispras.ru/wp-content/uploads/2012/07/

11. Ing-Jr Ding, Developments of Machine Learning Schemes for Dynamic Time-Wrapping-Based Speech Recognition // Ing-Jr Ding, Chih-Ta Yen, Yen-Ming Hsu. Mathematical Problems in Engineering. 2013.

12. Ramage, D. Hidden Markov Models Fundamentals // Daniel Ramage. CS229 Section Notes. 2007.

13. Система исследования речевых компонентов В.С. Шерхонов [Электронный ресурс] – режим доступа: http://www.stelani.ru/services/uslugi-po-napravleniyu-rechevye-tekhnologii/350/

14. [Электронный ресурс] – режим доступа: http://www.stel.ru/services/uslugi-po-napravleniyu-rechevye-tekhnologii/460/

15. [Электронный курс] – режим доступа: http://seminar.at.ispras.ru/wp-content/uploads/2012/07/Идентификация-диктора-по-голосу-текст

16. Bilmes, A. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models // *A. Jeff.* Berkley, C. A: International Computer Science Institute. 1998. P. 7–13.

17. [Электронный курс] – режим доступа: https://research-journal.org/technical/ispolzovanie-preobrazovaniya-gilberta-xuanga-dlya-formirovaniya-modelej-fonem-russkogo-yazyka-v-zadache-raspoznavaniya-rechi/

18. Дашкевич, И. В. Использование Вейвлет-преобразования в задаче голосовой идентификации диктора. / И. В. Дашкевич, М. С. Медведев // Международная научно-практическая конференция «Новшества в области технических наук». – Тюмень; Секция №20.

19. [Электронный курс] – режим доступа: seminar.at.ispras.ru/wp-content/.../Идентификация-диктора-по-голосу-текст

20. [Электронный курс] – режим доступа: https://cloud.google.com/text-to-speech/docs

21.[Электронный курс] – режим доступа: https://tech.yandex.ru/speechkit/mobilesdk/doc/ios/2.5/ref/group__vocalizer__globals__group__vc-docpage/

ПРИЛОЖЕНИЕ А

Листинг wav_param.m

```
function [P,segnames] = wav_param(Y,phonames,levelwavelet,typewavelet)

xds=Y'

Fram=800;
Nfr=1; Pt=[];
cca=0;
xds=mapminmax(xds);
for ff=1:fix(length(xds)/Fram)
    Nxds=xds(Nfr:Nfr+Fram);



    [c,l]=wavedec(Nxds,levelwavelet,typewavelet);
    DCELL=detcoef(c,l,'cells');

    for kk=1:10
    NRG(kk)= sum(abs(DCELL{kk}))';
    end

    Pt(:,ff)=[NRG]';
    Nfr=Nfr+Fram+1;
end

cca=cca+1;
if cca>1
    P=[P,Pt];
else    P=Pt;
end;



segnames=0;
```

Листинг net_train.m

```
function [net] = net_train(WDS,S1,net_goal)
WDS=getappdata(gcf,'WDS')

alphab='';
count=0; cc=0;

ff=size(WDS,2);
WDS2=struct();

for ii=1:ff
    if ii>1
    WDS2(ii-1).word=WDS(ii).word;
    WDS2(ii-1).fts=WDS(ii).fts;
    end
end;

WDS=WDS2;
ff=ff-1;

for ii=1:ff
   cc=WDS(ii).word;
   have=0;

   for jj=1:size(alphab,1)

       l1= alphab(jj,:)
        if strcmp(alphab(jj,:),cc);
           have=1 ;
        end;
    end;
if (have==0)
    count=count+1;
    alphab=strvcat(alphab,cc);
end;
end;
count



V=zeros(count,count);

for ii=1:count
        V(ii,ii)=1;
end;

T=zeros(count,length(2));

for ii=1:size(WDS,2)
    for jj=1:count

        if (strncmp(WDS(ii).word,alphab(jj,:),size(alphab(jj,:),1)))

            T(jj,ii)=1;
        end;
    end;
```

```
end;

    Pm=250;
    P=zeros(250,ff);
    temp=0; dm=0; ztemp=0;

    for ii=1:size(WDS,2)
    nframes=size(WDS(ii).fts);
    nframes=nframes(2)*10;
    temp=reshape(WDS(ii).fts,1,nframes);
    temp=temp(:);
        dm=Pm-size(temp,1);
        temp=[temp;zeros(dm,1)]

    P(:,ii)=temp;
end
    P

 pr=minmax(P);

S1= 250;
S2= 100;
S3=count;

net = newff( pr , [S1 S2 S3 ]  ,{'logsig' 'logsig' 'logsig'},'trainrp');

net.performFcn = 'sse';
net.trainParam.goal = net_goal;
net.trainParam.show = 20;
net.trainParam.epochs = 10000;
net.trainParam.mc = 0.95;

net = init(net);
[net,tr] = train(net,P,T);
net.userdata=alphab
```

Листинг recognize.m

```
function [phonems] = recogniz(masW,fs,net,diction,levelwavelet,typewavelet)

cnt=1;
Fram=800;
Nfr=1;
alphabet=net.userdata;
P=[]; Pm=250;
xds=masW';
Ln=length(masW');

xds=mapminmax(xds);
       while Nfr<=(Ln-Fram)

           Nxds=xds(Nfr:Nfr+Fram);
           [c,l]=wavedec(Nxds,levelwavelet,typewavelet);

            DCELL2=detcoef(c,l,'cells');
            Nfr=Nfr+Fram+1;



            for kk=1:10
                NRG2(kk,1)= sum(abs(DCELL2{kk}))';



            end
            NRG2;
            P=[P;NRG2];

       end
           dm=Pm-size(P,1);
           P=[P;zeros(dm,1)]

           Q(:,cnt)=sim(net,P)
           [M,ind] = max(Q(:,cnt)');

           plot(P)

 outword=alphabet(ind,:)

H=text_out;
data = guidata(gcf);
set(data.edit1,'string',outword);

end
```

Листинг wav.m

```
function varargout = wav(varargin)

if nargin == 0  % LAUNCH GUI

fig = openfig(mfilename,'reuse');

mysqldb.server = 'localhost';
mysqldb.user = 'root';
mysqldb.password = '';
mysqldb.db = 'rech';

global polzovatel;
polzovatel = 'nobody';

setappdata(fig, 'server', mysqldb.server );
setappdata(fig, 'user',   mysqldb.user );
setappdata(fig, 'password', mysqldb.password);
setappdata(fig, 'sqldb', mysqldb.db);

Y=0; fs=44100; nbits=16;
struct.inPoint=0;
struct.outPoint=1;
vops={10,'db8'};
fops={2048,'hamming',0.75};
net_opt=[20 , 1];
% инициализация данных приложения
setappdata(fig,'sampler',Y);      % текущий обрабатываемый сигнал
setappdata(fig, 'fd',fs);
setappdata(fig, 'nbits',nbits);   % разрядность
setappdata(fig, 'theAudioRecorder',audiorecorder(fs,nbits,1)); % объект
записи

setappdata(fig, 'theAudioPlayer',audioplayer(Y,fs));        % объект воспр.

setappdata(fig,'audioSelection',struct);
setappdata(fig,'samples',struct);
segments=0;
samples=struct();
setappdata(fig,'segments',segments);% сегменты обрабатываемого сигнала
setappdata(fig,'fft_opt',fops);     % ячейка с параметрами
setappdata(fig,'net',Y);           % нейросеть
traindata=0;
setappdata(fig,'traindata',traindata); %обучающая выборка
phonames='';
setappdata(fig,'phonames',phonames);
setappdata(fig,'net_opt',net_opt);      % параметры сети
setappdata(fig,'wav_param_opt',vops);
S={};
```

```
setappdata(fig,'S',S);

diction={};
setappdata(fig,'diction',diction); %словарь
%отображение панели управления (запись, воспр, пауза, стоп)
[htoolbar, haudiobtns]=render_audiotoolbar(fig);



    handles = guihandles(fig);
    guidata(fig, handles);

    data = guidata(fig);
    if nargout > 0
        varargout{1} = fig;
    end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end

end;



function varargout = pushbutton1_Callback(h, eventdata, handles,varargin )
v=0;
% определение количества входных параметров
if nargin > 3
v=varargin{1};
end

if v~=1
v=0;
end

% Вызов текущего сигнала
Y=getappdata(gcbf,'sampler');
player=getappdata(gcf,'theAudioPlayer');
FS=get(player,'SampleRate');
NBITS=get(player,'BitsPerSample');

% определение установленных параметров
fops=getappdata(gcbf,'fft_opt');
overlap = fops{3};
nfft=fops{1};
win=fops{2};
```

```
overlap = round(overlap*nfft);

% Передача входных параметров функции определения частоты основного тона
% и формантного анализа
[fosn,n]=span(Y,FS,nfft,win,overlap,v);

% --------------------------------------------------------------------
% Вызов функции сегментации
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)

% Вызов текущего сигнала
Y=getappdata(gcbf,'sampler');
player=getappdata(gcf,'theAudioPlayer');
FS=get(player,'SampleRate');
NBITS=get(player,'BitsPerSample');

% Вызов функции сегментации
[s1,C2] = words(Y,FS,NBITS);

C3(1)={Y};

% отображение полученных сегментов в виде списка
data = guidata(gcf);
name='сегмент';
names='речевой сигнал';

for i=1:s1
names=strvcat(names,strcat(name,int2str(i)));
C3(i+1)=C2(i);
end
set(data.listbox2,'string',names);

% сохранение результата сегментации в виде данных приложения
setappdata(gcf,'segments',C3);

% --------------------------------------------------------------------
% Включение функции увеличения и разметки для графика отображения сигнала
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
data = guidata(gcbf);
zoom xon;
grid on;



% меню Файл
function File_Callback(hObject, eventdata, handles)
function Open_Callback(hObject, eventdata, handles)

% запуск диалогового окна открытия файла
[file,path,filt]=uigetfile('*.wav','wave');
```

```
% передача данных файла объекту воспоизведения
[Y,FS,NBITS]=wavread(cat(2,path,file));
rmappdata(gcf,'theAudioPlayer');
setappdata(gcf,'theAudioPlayer',audioplayer(Y, FS, NBITS));

% установка нового значения текущего обрабатываемого сигнала
rmappdata(gcf,'sampler');
setappdata(gcf,'sampler',Y);
C3(1)={Y};

rmappdata(gcf,'segments');
setappdata(gcf,'segments',C3);

% отображение сигнала на временном графике
% рассчет длительности сигнала, с
data = guidata(gcf);
set(data.text4,'string',length(Y)/FS);
set(data.listbox2,'value',1);
set(data.listbox2,'string','речевой сигнал');
T=[0:(length(Y)-1)];
T=T'/FS;
plot(T,Y);

% --------------------------------------------------------------------
% меню Параметры
function Options_Callback(hObject, eventdata, handles)

%Вызов меню настройки параметров записи
function Record_Callback(hObject, eventdata, handles)

H = rec_opt;

function SetRecParam(fdis,numbit)


% --------------------------------------------------------------------
%Функция сохранения данных в виде звукового файла *.wav
function Save_as_Callback(hObject, eventdata, handles)


% запуск диалогового окна сохранения файла
[FILENAME, PATHNAME, FILTERINDEX]=uiputfile('*.wav', 'wav1');

 data = guidata(gcf);


 Y=getappdata(gcbf,'sampler');
 player=getappdata(gcf,'theAudioPlayer');
 FS=get(player,'SampleRate')
 NBITS=get(player,'BitsPerSample')
 wavwrite(Y,FS,NBITS,cat(2,PATHNAME,FILENAME));
```

```
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



% --- Executes on selection change in listbox2.
%Функция выбора сегмента из списка-----------------------------------
function listbox2_Callback(hObject, eventdata, handles)

n=get(hObject,'Value');
segments=getappdata(gcf,'segments');

player=getappdata(gcf,'theAudioPlayer');
FS=get(player,'SampleRate')

%построение временой диаграмы выбранного сигнала
T=[0:(length(segments{n})-1)];
T=T'/FS;
plot(T,segments{n});
%загрузка выбранного сигнала в объект воспроизведения
NBITS=get(player,'BitsPerSample')
rmappdata(gcf,'theAudioPlayer');
setappdata(gcf,'theAudioPlayer',audioplayer(segments{n}, FS, NBITS));
rmappdata(gcf,'sampler');
setappdata(gcf,'sampler',segments{n});
data = guidata(gcf);
set(data.text4,'string',length(segments{n})/FS);



% --------------------------------------------------------------------
%Выход из программы
function Exit_Callback(hObject, eventdata, handles)

selection = questdlg(['Закрыть ' get(handles.figure1,'Name') '?'],...
                     ['Закрыть ' get(handles.figure1,'Name') '...'],...
                     'Да','Нет','Да');
if strcmp(selection,'Нет')
    return;
```

```
end
close all;


% --------------------------------------------------------------------
function Help_Callback(hObject, eventdata, handles)

function about_Callback(hObject, eventdata, handles)

about;


% --------------------------------------------------------------------
% Вызов меню установки параметров БПФ
function fft_set_Callback(hObject, eventdata, handles)

H = fft_opt;

function mysql_set_Callback(hObject, eventdata, handles)

M = mysql_settings;

% Вызов меню установки пользователя
function polzovatel_set_Callback(hObject, eventdata, handles)

U = select_user;


% --- Executes on button press in pushbutton4.
%Запуск формантного анализа
function pushbutton4_Callback(hObject, eventdata, handles)

pushbutton1_Callback(gcbf,0,0,1)


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)

Y=getappdata(gcbf,'sampler');
player=getappdata(gcf,'theAudioPlayer');
FS=get(player,'SampleRate');

% определение установленных параметров БПФ
fops=getappdata(gcbf,'fft_opt');
overlap = fops{3};
nfft=fops{1};
win=fops{2};
overlap = round(overlap*nfft);

phonem(Y,FS,nfft,win,overlap);
```

```
%--------------------------------------------------------------------------
function Import_Net_Callback(hObject, eventdata, handles)

[file,path,filt]=uigetfile('*.mat','Net');

% передача данных файла объекту воспоизведения
nnet=load(cat(2,path,file));
temp = struct2cell(nnet);
nnet=temp{1}
rmappdata(gcf,'net');
setappdata(gcf,'net',nnet);

%--------------------------------------------------------------------------
% Сохранение сети
function Export_Net_Callback(hObject, eventdata, handles)

[FILENAME, PATHNAME, FILTERINDEX]=uiputfile('*.mat', 'mat1');

data = guidata(gcf);
NET=getappdata(gcf,'net');
fpath=cat(2,PATHNAME,FILENAME);
save(fpath, 'NET');

%--------------------------------------------------------------------------
% распознавание
function pushbutton10_Callback(hObject, eventdata, handles)

Y=getappdata(gcbf,'sampler');
player=getappdata(gcf,'theAudioPlayer');
FS=get(player,'SampleRate');
    vops=getappdata(gcf,'wav_param_opt');
    levelwavelet = vops{1};
    typewavelet = vops{2};
NET=getappdata(gcf,'net');
diction=getappdata(gcf,'diction');
recogniz(Y,FS,NET,diction,levelwavelet,typewavelet);

%--------------------------------------------------------------------------
function listbox3_Callback(hObject, eventdata, handles)


function pushbutton11_Callback(hObject, eventdata, handles)

H= phoname;

%--------------------------------------------------------------------------
% Вычисление признаков
function pushbutton8_Callback(hObject, eventdata, handles)

P=getappdata(gcf,'traindata');  % получение обучающей выборки
net_opt=getappdata(gcf,'net_opt')
phonames=getappdata(gcf,'phonames');
```

```
data = guidata(gcf);
nnet=net_train(phonames,P,net_opt(1),net_opt(2));                % обучение
сети
%close;
rmappdata(gcf,'net');
setappdata(gcf,'net',nnet);                % сохранение сети

%--------------------------------------------------------------------
% очистка обучающей выборки
function pushbutton9_Callback(hObject, eventdata, handles)

data = guidata(gcf);
set(data.listbox3,'string','');
rmappdata(gcf,'traindata');
traindata=0;
setappdata(gcf,'traindata',traindata);

%--------------------------------------------------------------------
% Выделение из речевого сигнала
function pushbutton12_Callback(hObject, eventdata, handles)

data = guidata(gcf);
Y=getappdata(gcf,'sampler');
C=getappdata(gcf,'segments');
player=getappdata(gcf,'theAudioPlayer');
FS=get(player,'SampleRate')

names=get(data.listbox2,'string');
time_range=get(data.axes2,'XTick')

N1=fix(time_range(1)*FS)
N2=fix(time_range( numel(time_range) )*FS)

ph_sample=Y(N1:N2);
L=numel(C)

C{L+1}=ph_sample;
set(data.listbox2,'string',strvcat(names,'сегмент'));
setappdata(gcf,'segments',C);

%----- Запуск редактора ------------------------------------------
function Phon_edit(hObject, eventdata, handles)
data = guidata(gcf);
S=getappdata(gcf,'S')
H= phon_edit;
data=guidata(gcf);
set(data.uitable1,'data',S);

%----- Подключение сочетаний -------------------------------------
function Phon_open(hObject, eventdata, handles)
S=load('static_rus.mat');
S=getfield(S,'S');
rmappdata(gcf,'S');
```

```
setappdata(gcf,'S',S);
H=phon_connect;

%----- Сохранение ----------------
function Phon_save(hObject, eventdata, handles)
data = guidata(gcf);
S=getappdata(gcf,'S');
fpath='static_rus.mat';
save(fpath, 'S');
H=phon_save;

%----- Запуск редактора словаря -------------------------------------------
function Diction_edit(hObject, eventdata, handles)
data = guidata(gcf);
diction=getappdata(gcf,'diction')
H= dict_edit;
data = guidata(gcf);
set(data.listbox1,'string',diction);


%----- Подключение словаря  сохранение словаря ----------------------------
--------------
function Dict_open(hObject, eventdata, handles)

[file,path,filt]=uigetfile('*.mat','Dict');

diction=load(cat(2,path,file));
diction=getfield(diction,'diction');
rmappdata(gcf,'diction');
setappdata(gcf,'diction',diction);


%----- Сохранение словаря -------------------------------------------
function Dict_save(hObject, eventdata, handles)
[FILENAME, PATHNAME, FILTERINDEX]=uiputfile('*.mat', 'mat1');

data = guidata(gcf);
diction=getappdata(gcf,'diction');
fpath=cat(2,PATHNAME,FILENAME);
save(fpath, 'diction');

%---------------- Сохранить выборку
function Par_save(hObject, eventdata, handles)

[FILENAME, PATHNAME, FILTERINDEX]=uiputfile('*.mat', 'parameters');

data = guidata(gcf);
traindata=getappdata(gcf,'traindata');
phonames=getappdata(gcf,'phonames');

params{1}=traindata;
params{2}=phonames;
```

```
fpath=cat(2,PATHNAME,FILENAME);
save(fpath, 'params');

%------------------- Загрузить выборку
function Par_open(hObject, eventdata, handles)

[file,path,filt]=uigetfile('*.mat','Parameters');
data = guidata(gcf);

params=load(cat(2,path,file));
params=getfield(params,'params');
rmappdata(gcf,'traindata');
rmappdata(gcf,'phonames');
setappdata(gcf,'traindata',params{1});
setappdata(gcf,'phonames',params{2});

set(data.listbox3,'string',params{2});


%---------------- Экспорт списка сигналов ------------------

function Sample_export(hObject, eventdata, handles)
Y=0;
sname='';
path='';

folder=uigetdir('')
data = guidata(gcf);

samples=getappdata(gcf,'samples')
snum=size(samples,2)

for nn=1:snum
    Y=samples(nn).sample;
    sname=samples(nn).name;

    path=strcat(folder,'\',sname,'_',num2str(nn),'.wav');

    wavwrite(Y,22050,16,path);
end



function Sample_import(hObject, eventdata, handles)
Y=0;
sname='';
phname='';
path='';
sample=struct();
names='';
P=0;
folder=uigetdir('');
data = guidata(gcf);
```

```
list3names='';

d=dir(cat(2,folder,'\*.wav'));
snum=size(d,1);

for nn=1:snum
    sname=d(nn).name;
    phname = regexp(sname, '_', 'split');
    path=cat(2,folder,'\',sname);
    Y=wavread(path);
    samples(nn).name=char(phname(1));
    samples(nn).sample=Y;

   names=strvcat(names,sname);
   C3(nn)={Y};


   [Pt,segnames]=wav_param(Y,char(phname(1)),6,'db8');
   if P==0
    P=Pt;

    phonames=segnames;
    else
    P=[P,Pt];

    phonames=strvcat(phonames,segnames);
    end;

    if length(list3names)==0
    list3names=char(phname(1));
    else
    list3names=strvcat(list3names,char(phname(1)));
    end


end

rmappdata(gcf,'traindata');
setappdata(gcf,'traindata',P);
rmappdata(gcf,'phonames');
setappdata(gcf,'phonames',phonames);
set(data.listbox3,'string',list3names);

setappdata(gcf,'samples',samples);
set(data.listbox2,'string',names);
setappdata(gcf,'segments',C3);



function Segment_import(hObject, eventdata, handles)
Y=0;
sname='';
```

```
phname='';
path='';
sample=struct();
names='';
P=0;
folder=uigetdir('');
data = guidata(gcf);
list3names='';

%d=dir(folder)
d=dir(cat(2,folder,'\*.wav'));
snum=size(d,1);

for nn=1:snum
    sname=d(nn).name;
    phname = regexp(sname, '_', 'split');
    path=cat(2,folder,'\',sname);
    Y=wavread(path);
    samples(nn).name=char(phname(1));
    samples(nn).sample=Y;

   names=strvcat(names,sname);
   C3(nn)={Y};

   end;



setappdata(gcf,'samples',samples);
set(data.listbox2,'string',names);
setappdata(gcf,'segments',C3);

function test_Callback(hObject, eventdata, handles)

data = guidata(gcf);
C3=getappdata(gcf,'segments');
phonames=get(data.listbox2,'string');
n_net=getappdata(gcf,'net');
ph_test(C3,phonames,n_net)




%-------------------------------------------------------------
function net_opt(hObject, eventdata, handles)

H= net_opt;


%------------------Коннектимся к БД----------------------------
function use_mysql_db(hObject, eventdata, handles)
```

```
mysql('open', getappdata(gcf, 'server'), getappdata(gcf, 'user'),
getappdata(gcf, 'password'));
mysql('use', getappdata(gcf, 'sqldb'));

function close_mysql_db(hObject, eventdata, handles)
mysql('closeall');

function check_query(hObject, eventdata, handles)
global polzovatel;
query = strcat(strcat('SELECT dir,file_name FROM files_words WHERE (user_id =
(SELECT id FROM users WHERE (fio = "',polzovatel), '")))');
mysql(query);

function download_query(hObject, eventdata, handles)
F = ftp_download;

function wav_param_opt_Callback(hObject, eventdata, handles)
%
H = wav_param_opt;
```